UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE CIENCIAS MATEMÁTICAS

TESIS DOCTORAL

Una lógica para programación lógica

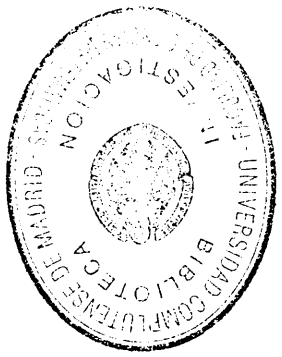
MEMORIA PARA OPTAR AL GRADO DE DOCTOR
PRESENTADA POR

Juan José Moreno Navarro

Madrid, 2015

FACULTAD DE CIENCIAS MATEMATICAS

UNIVERSIDAD COMPLUTENSE DE MADRID



UNA LOGICA PARA PROGRAMACION LOGICA

Juan José Moreno Navarro

Dirigido por el Prof. Mario Rodríguez Artalejo

Trabajo presentado por el autor en la Facultad de Ciencias Matemáticas de la Universidad Complutense de Madrid para acceder al Grado de Licenciado.

Septiembre 1985

No por obligado es menos sentido mi agradecimiento a Mario Rodríguez Artalejo por la enorme ayuda brindada para la realización de éste trabajo.

Es, además, larga la lista de gente que me ha apoyado (y soportado) durante su elaboración. Aunque a estos últimos no los cite explicitamente:

A todos ellos ¡Gracias¡.

La inter - vida manejaba abstracciones tales como espíritu y conciencia, que la para - vida escuchaba como quien oye llover - tarea delicada. Por supuesto, la infra - vida pedía a cada instante el queso rallado, y la super - vida trinchaba el pollo en cuarenta y dos movimientos, método Stanley - Fitzsimmons.

Julio Cortázar

(Historias de cronopios y famas).

INDICE:

INTRODUCCION 5

Capítulo 1.- LOGICA DINAMICA STANDARD PARA PROGRAMAS REGULARES

1.1.- Introducción. Antecedentes históricos 10

1.2.- Definición de DL 12

1.3.- Capacidad de expresión de DL 16

1.4.- Axiomatización infinitaria de DL. El cálculo IDL.Teorema de completitud 22

1.5.- Estructuras aritméticas 45

1.6.- El cálculo P. Teorema de completitud 49

Capítulo 2.- LOGICAS DE LA PROGRAMACION CON SEMANTICAS NO STANDARD

2.1.- Introducción 59

2.2.- La lógica NDL. Definición 62

2.3.- El cálculo NP. Teorema de completitud 68

2.4.- Potencia de NDL. Comparación con otras lógicas de la programación 74

2.5.- Una lógica de la programación de primer orden para programas recursivos 95

Capítulo 3.- UNA LOGICA NO STANDARD PARA PROGRAMAS LOGICOS

3.1.- Introducción: La lógica NPL para programación lógica 108

3.2.- Definición de NPL. El cálculo LNP.

Teorema de completitud 111

3.3.- Estructuras standard y estructuras admi-

sibles. Semántica de programas lógicos sim

ples sobre estructuras standard y estructu

ras admisibles 124

3.4.- Potencia de NPL. Aplicación práctica de

NPL a la derivación formal de propieda-

des de programas lógicos. 152

BIBLIOGRAFIA 176

INTRODUCCION

El diseño de un programa que resuelva un cierto problema, para una máquina matemática o real, lleva asociado el problema de su corrección. La verificación de un programa es un problema complejo en ciertos casos, y que, en general, se resuelve más por la intuición y el "convencimiento" del diseñador, que usando un método general y adecuado.

El uso de la lógica en este problema es una vieja idea ya expresada por Turing (44) y McCarthy (30), cuyos primeros trabajos se deben a Naur (32) y Floyd (14) y que ya fueron formalizados por Hoare (22) y Dijkstra (10). Surgen asi las lógicas de la programación. Probablemente la lógica de Hoare es el formalismo más popular para probar la corrección parcial de programas, aunque se ha establecido su incompletitud.

Una precisa formulación de un sistema lógico de razonamiento acerca de programas fue dada por Pratt (36), basandose en la lógica modal y en los trabajos de Engeler (12) y Salwicki (39), y posteriormente desarrollada por Harel (18) y otros, y se le conoce por el nombre de Lógica Dinámica. Básicamente los programas se integran en un lenguaje de aserciones y se tratan como operadores modales en la categoría de las fórmulas, que expresan propiedades de los programas.

Para la lógica dinámica solo se consiguen teoremas de completitud con cálculos infinitarios a causa del operador de iteración $*$.

Desde el punto de vista de la implementación de sistemas de ayuda a la verificacion estos cálculos son poco útiles.

Dentro de la lógica dinámica solo es posible obtener cálculos finitarios completos para ciertas estructuras. El teorema de completitud de Harel (18) demuestra la completitud de un cálculo finitario en estructuras aritméticas (estructuras que contienen a los números naturales). Los números naturales y sus operaciones son utilizados para "contar" el numero de iteraciones de un programa. Este teorema generaliza el conocido teorema de completitud de Cook para la lógica de Hoare sobre estructuras expresivas.

En todo caso este resultado presenta el inconveniente de necesitar como axiomas toda la estructura de primer orden de la estructura con la que se trabaje, y estas teorías son altamente indecidibles.

Con el fin de evitar estos problemas algunos autores han propuesto alterar la semántica de los programas. Estas semántica "no standard" afectan a la iteración. Si admitimos modelos no standard de la teoría de números se pueden obtener iteraciones, finitas en nuestras estructuras, pero que son intuitivamente infinitas.

Hajek (17), con estas ideas, enunció y probó un teorema de completitud semejante al de Harel sobre modelos de la aritmética de Peano. La lógica resultante es recursivamente axiomatizable.

Aunando estas ideas con otras de la lógica temporal de programas, Andréka, Németi,y Sain desarrollaron independientemente la Lógica Dinámica No Standard. Las estructuras con las que trabaja son de tres géneros e incluyen, además de la estructura de datos (objetos que manejan los programas) una estructura temporal (los distintos instantes de la ejecución de un programa) y un conjunto de sucesiones (de estados de cómputo a lo largo del tiempo).

A esta lógica se la puede dotar de un cálculo completo. Además de ofrecer un formalismo adecuado para razonar acerca de propiedades de programas esta lógica puede emplearse para investigar las relaciones entre otras lógicas de programas.

Otro ejemplo de una lógica que usa semántica no standard es la lógica de primer orden de Cartwright (6) para programas recursivos. Las ideas no standard y en particular la noción de mínimo punto fijo definible, permiten conciliar la semántica de punto fijo y la lógica de primer orden, pese a las conocidas críticas de Hitchock y Park(21).

Estas lógicas permiten concluir que la semántica no standard es la que se corresponde adecuadamente con el empleo de formalismos de primer orden para derivar propiedades de programas.

Dada la importancia y el interes de la programación lógica en los últimos años hemos desarrollado en el presente trabajo una lógica no standard para el razonamiento acerca de propiedades de programas lógi

cos que amplian los conocidos programas de Horn sobre interpretaciones de Herbrand.

Esta lógica toma ideas de los trabajos de Andréka, Németi y Sain usando sus estructuras de tres géneros (las sucesiones codifican cadenas de fórmulas que representan deducciones, que se pueden imaginar como cómputos de programas lógicos) y de los de Cartwright en lo referente a mínimos puntos fijos definibles.

Se prueba que ésta lógica es completa, y que permite simular los programas de la lógica dinamica y los programas recursivos. Además los programas lógicos pueden considerarse como conjunto de fórmulas de primer orden lo que les da una semántica declarativa natural, heredada de la lógica. Estos resultados generalizan los obtenidos por van Emdem - Kowalski (11) y Apt - van Emdem (3) para programas de Horn e interpretaciones de Herbrand.

El trabajo está organizado como sigue:

En el capítulo 1 se presenta la Lógica Dinámica y se prueba un teorema de completitud para un cálculo infinitario (con una demostracion propia basada en la idea de Schmitt (40) de sumergir la lógica dinámica en la lógica $L_{\omega_1\omega}$) y el teorema de completitud de Harel para estructuras aritméticas.

El capítulo 2 se ocupa de algunas lógicas de la programación con semántica no standard, investigadas en los últimos años por varios

autores. Se estudiará la lógica dinámica no standard de Andréka, Németi y Sain en particular el cálculo completo y la caracterización dentro de esta lógica del cálculo de Harel para lógica dinámica (ésta demostración desarrolla el brevísimo esquema de la prueba que ofrece Sain en (38)).

También se presenta la lógica de primer orden para programas recursivos de Catwright adaptada a nuestro contexto, consiguiendo así una simplificación en la exposición y facilidad en el trabajo.

En el capítulo 3 presentamos nuestra lógica no standard para programas lógicos. Este capítulo es el núcleo del trabajo, aunque los capítulos anteriores sirven para ver los antecedentes históricos de lo trabajado en el tema y detallar los estudios en los que nos basamos. Demostramos los resultados antes mencionados: Existencia de un cálculo finitario completo y equivalencia entre tres caracterizaciones semánticas, empleando las ideas de estructura admisible y de mínimo punto fijo definible.

La última sección es de caracter más aplicado. En ella presentamos la metodología propuesta por Clark y Tärnlud (7) para la especificación, diseño y verificación de programas lógicos, y mostraremos por medio de ejemplos que las pruebas de corrección de programas empleadas por estos autores pueden ser formalizadas en el marco de nuestra lógica.

CAPITULO 1: LOGICA DINAMICA STANDARD PARA PROGRAMAS REGULARES.

1.1.- Introducción. Antecedentes históricos.

La lógica dinámica es una formulación precisa de un sistema lógico de razonamiento acerca de programas. Fue enunciada por Pratt(36), basandose en la lógica modal. Está emparentada con la lógica de programas de Salwicki(39) y fue posteriormente desarrollada por Harel(18). Su idea es integrar los programas en un lenguaje de aserciones, tratando los programas como operadores modales en la categoría de las fórmulas, las cuales expresan propiedades de los programas.

El trabajo, ya clásico, de Fischer y Ladner (13) sobre lógica dinámica proposicional, fue punto de partida para los estudios en esta lógica, proseguidos por otros investigadores(cfr. (34) y (19)).

La lógica dinámica proposicional PDL mantiene con la lógica dinámica de primer orden DL la misma relación que el cálculo proposicional tiene con la lógica de primer orden. El problema de validez de PDL es decidible, mientras que el de DL es Π_1^1, por lo que no es recursivamente axiomatizable.

La lógica dinámica de primer orden DL es definida en 1.2 y su capacidad de expresión es discutida en 1.3.

Debido al operador de iteración * se requieren cálculos infinitarios para obtener teoremas de completitud para esta lógica. Un cálcu-

lo infinitario IDL (propuesto por Harel en (18)) es presentado en 1.4, y demostrada su completitud mediante una inmersión de DL en la lógica $L_{\omega_1\omega}$ (segun idea de Schmitt en (40)).

Desde el punto de vista de la implementación de estos sistemas para ayuda a la verificación son deseables, sin embargo, cálculos efectivos y finitarios.

Estos teoremas de completitud con cálculos finitarios solo son conseguidos en forma relativa, es decir, para ciertas estructuras. El teorema de completitud aritmética de Harel (cfr: (18)) prueba la completitud en estructuras aritméticas (definidas en 1.5) del cálculo P. Este cálculo y el teorema y su demostración pueden encontrarse en 1.6, así como sus relaciones con el conocido teorema de completitud relativa de Cook para la lógica de Hoare (cfr. (22) y (9)).

Sin embargo, estos resultados siguen presentando el inconveniente de que la teoría de primer orden de una estructura aritmética es altamente indecidible. En capítulos posteriores mostraremos que es posible dotar a la lógica dinámica de un cálculo finitario completo, si bien a costa de alterar la semántica de los programas.

1.2.- Definición de DL.

DL es la lógica dinámica de primer orden para programas regulares.

Sintaxis de DL:

Trabajaremos con un tipo de similaridad dado d (llamado tipo de similaridad de los datos) que incluye una serie de símbolos de función $f^{(n)}$ y de símbolos de predicado $P^{(n)}$ entre los que suponemos se haya la igualdad "$\doteq$".

Se definen recursivamente los términos de tipo $d: TS_d$, las fórmulas de primer orden de tipo $d: FS_d$, los programas regulares de tipo d PS_d y las fórmulas dinámicas de tipo $d: DFS_d$.

Definición 1.2.1: TS_d, FS_d, PS_d, DFS_d

TS_d

$c \in d$ (c constante) $\Rightarrow$ $c \in TS_d$

$x \in V$(conjunto de variables del lenguaje) $\Rightarrow$ $x \in TS_d$

$$\left.\begin{array}{l} \tau_1, \ldots, \tau_n \in TS_d \\ f^{(n)} \in d \end{array}\right\} \Rightarrow f(\tau_1, \ldots, \tau_n) \in TS_d$$

FS_d

true, false FS_d

$\tau, \sigma \in TS_d \Rightarrow \tau \doteq \sigma \in FS_d$

$$\left.\begin{array}{l} P^{(n)} \in d \\ \tau_1, \ldots, \tau_n \in TS_d \end{array}\right\} \Rightarrow P(\tau_1, \ldots, \tau_n) \in FS_d$$

$\varphi, \psi \in FS_d$, $x \in V \Rightarrow \neg\varphi$, $(\varphi \wedge \psi)$, $\exists x \varphi \in FS_d$

PS_d y DFS_d

$x \in V$ y $\tau \in TS_d \Rightarrow (x:=\tau) \in PS_d$

$\alpha, \beta \in PS_d \Rightarrow (\alpha;\beta), (\alpha \cup \beta), (\alpha^*) \in PS_d$

$\varphi \in DFS_d \Rightarrow \varphi? \in PS_d$

$\varphi \in FS_d \Rightarrow \varphi \in DFS_d$

$\varphi, \psi \in DFS_d$, $x \in V$, $\alpha \in PS_d \Rightarrow \neg\varphi, (\varphi \wedge \psi), \exists x \varphi, \langle\alpha\rangle \varphi \in DFS_d$

Abreviaturas:

$\varphi \vee \psi = \neg(\neg\varphi \wedge \neg\psi)$

$\forall x \varphi = \neg \exists x \neg\varphi$

$[\alpha] \varphi = \neg\langle\alpha\rangle\neg\varphi$

Observación: Los programas PS_d son programas no deterministas. $(\alpha;\beta)$ corresponde a la idea de concatenación de programas, $(\alpha \cup \beta)$ supone la ejecución de α o β de forma no determinista, $\varphi?$ supone una pregunta: si la fórmula φ es cierta el programa continua, si no el programa falla, α^* es la iteración no determinista del programa α .

La fórmula $\langle\alpha\rangle \varphi$ se lee " es posible despues de ejecutar α que se cumpla φ " y por dualidad $[\alpha] \varphi$ significa " despues de ejecutar α necesariamente se cumple φ ".

Esta idea intuitiva es lo que expresa la semántica de programas y fórmulas de DL.

Semántica de DL:

Definición 1.2.2.- STR_d

STR_d es la clase de las estructuras de tipo de similaridad d

$\mathfrak{D} = (D , (f^{\mathfrak{D}})_{f \in d}, (P^{\mathfrak{D}})_{P \in d})$

con $f^{\mathfrak{D}}: D^n \longrightarrow D$ si $f = f^{(n)}$

$P^{\mathfrak{D}}: D^n \longrightarrow \{0,1\}$ si P es $P^{(n)}$

La idea es que $\mathfrak{D}$ es el tipo de datos o la estructura de datos.

Definición 1.2.3.- Estados sobre $\mathfrak{D}$: $W_{\mathfrak{D}}$

D^V es el conjunto de estados sobre $\mathfrak{D}$. Llamamos $W_{\mathfrak{D}}$ a los estados sobre $\mathfrak{D}$ ($W_{\mathfrak{D}} = D^V$) y los elementos $s,t,r... \in W_{\mathfrak{D}}$ son aplicaciones que asignan valores en D a las variables.

Por inducción definiremos simultaneamente $\varphi^{\mathfrak{D}}$, el significado de φ en $\mathfrak{D}$, y $\alpha^{\mathfrak{D}}$, significado de α en $\mathfrak{D}$.

$\varphi^{\mathfrak{D}}: W_{\mathfrak{D}} \longrightarrow \{0,1\}$ $\qquad \varphi \in DFS_d$

$\alpha^{\mathfrak{D}}: W_{\mathfrak{D}} \times W_{\mathfrak{D}} \longrightarrow \{0,1\}$ $\qquad \alpha \in PS_d$

y, por convenio escribiremos

$\mathfrak{D} \models_s \varphi \iff \varphi^{\mathfrak{D}}(s)=1$

$s \; \alpha^{\mathfrak{D}} \; t \iff \alpha^{\mathfrak{D}}(s,t)=1$

Definición 1.2.4.- Semántica de fórmulas y programas de DL

$\mathfrak{D} \models_s \tau \doteq \sigma \iff \tau^{\mathfrak{D}}(s) = \sigma^{\mathfrak{D}}(s)$

($\tau^{\mathfrak{D}}(s)$ se define de la manera habitual en lógica de primer orden)

$\mathfrak{D} \models_s P(\tau_1,.....,\tau_n) \iff P^{\mathfrak{D}}(\tau_1(s),.....,\tau_n(s))$

$\mathfrak{D} \models_s \underline{\text{true}}$, $\quad \mathfrak{D} \not\models_s \underline{\text{false}}$

$\mathfrak{D} \models_s \neg\varphi \iff \mathfrak{D} \not\models_s \varphi$

$\mathfrak{D} \models_s (\varphi \wedge \psi) \iff \mathfrak{D} \models_s \varphi$ y $\mathfrak{D} \models_s \psi$

$\mathfrak{D} \models_s \exists x \varphi \iff$ existe $a \in D$ tal que $\mathfrak{D} \models_{s\,[a/x]} \varphi$

$\mathfrak{D} \models_s \langle\alpha\rangle \varphi \iff$ existe $t \in W_{\mathfrak{D}}$ tal que $s\,\alpha^{\mathfrak{D}} t$ y $\mathfrak{D} \models_t \varphi$

$s\,(x:=\tau)^{\mathfrak{D}}\,t \iff t = s\,[\tau^{\mathfrak{D}}(s)/x]$

$s\,\varphi?^{\mathfrak{D}}\,t \iff \mathfrak{D} \models_s \varphi$ y $t=s$

$s(\alpha;\beta)^{\mathfrak{D}}\,t \iff$ existe $r \in W_{\mathfrak{D}}$ tal que $s\,\alpha^{\mathfrak{D}} r\,\beta^{\mathfrak{D}} t$

$s(\alpha \cup \beta)^{\mathfrak{D}}\,t \iff s\,\alpha^{\mathfrak{D}} t$ o $s\,\beta^{\mathfrak{D}} t$

$s(\alpha^*)^{\mathfrak{D}} t \iff$ existen $s=r_0, r_1, \ldots, r_n=t$ tal que $r_0 \alpha^{\mathfrak{D}} r_1 \alpha^{\mathfrak{D}} \ldots \alpha^{\mathfrak{D}} r_n$ $(n \geq 0)$

Definición 1.2.5

- φ es satisfactible sobre $\mathfrak{D}$: $\mathrm{Sat}^{\mathfrak{D}}\varphi \iff$ existe $s \in W_{\mathfrak{D}}$ tal que $\mathfrak{D} \models_s \varphi$
- φ es satisfactible : $\mathrm{Sat}\,\varphi \iff \mathrm{Sat}^{\mathfrak{D}}\varphi$ para algún $\mathfrak{D}$
- φ es válida en $\mathfrak{D}$: $\mathfrak{D} \models \varphi \iff \mathfrak{D} \models_s \varphi$ para todo $s \in W_{\mathfrak{D}}$
- φ es válida : $\models \varphi \iff$ para cualquier $\mathfrak{D}$. $\mathfrak{D} \models \varphi$
- φ es consecuencia de Φ : $\Phi \models \varphi \iff$ para cualquier $\mathfrak{D}$ $(\mathfrak{D} \models \Phi \Rightarrow \mathfrak{D} \models \varphi)$

1.3.- Capacidad de expresión de DL.

Tomemos $d = \{0, suc, +, \cdot, \leqslant\}$ y

$$\mathfrak{N} = (\, N,\ 0^N,\ suc^N,\ +^N,\ \cdot^N,\ \leqslant^N \,)$$

el tipo de similaridad de la aritmética y el modelo standar de la aritmética.

Usaremos libremente símbolos para funciones y relaciones definibles, como $>$ (relacion inversa de $<$), m.c.d. (máximo común divisor), etc.

Las siguientes fórmulas dinámicas son válidas en $\mathfrak{N}$

$\langle (x:=x\dot{-}1)^* \rangle\, x \doteq 0$ Todo número es standar

$y > 0 \vee \langle y=0? \rangle \underline{true}$ Todo número es positivo o 0

$(u \doteq x \wedge v \doteq y \wedge x \cdot y > 0) \longrightarrow \langle (u \neq v?; ((u > v?; u:=u-v) \cup (v > u?; v:=v-u)))^*; u=v? \rangle\ u \doteq m.c.d.(x,y)$

Corrección total de un programa que calcula el m.c.d. de dos números. Este programa es el habitual:

```
While u≠v do
        if  u > v  then  u:=u-v
                   else  v:=v-u   fi  od
```

Más en general, las construcciones *if* y *while* pueden expresarse en el lenguaje de los programas regulares como sigue:

if $\varphi(\bar{x})$ *then* α se expresa $((\varphi(\bar{x})?; \alpha) \cup (\neg\varphi(\bar{x})?; \beta))$

else β *fi*

suponiendo α y β ya traducidos.

While $(\bar{x})$ **do** **od** se expresa $((\varphi(\bar{x})?;\alpha)^* ; \neg\varphi(\bar{x})?)$

suponiendo α ya traducido.

Las siguientes fórmulas dinámicas son ejemplos de fórmulas válidas:

$[(x\doteq u \wedge y\doteq v)?;(u:=f(u) \cup v:=f(v))]\ (x\doteq u \vee y\doteq v)$

Una de las dos alternativas de $\cup$ no es ejecutada.

$x\doteq y \longrightarrow [(x:=f(f(x)))^*]\ \langle (y:=f(y))^* \rangle\ x\doteq y$

El proceso de iterar $f\circ f$ es un caso particular del proceso de iterar f.

$x\doteq y \longrightarrow [(x:=f(x))^*]\ (\varphi \longrightarrow (x\doteq y \vee \langle y:=f(y);(y:=f(y))^* \rangle\ \varphi[y/x])$

con y no libre en φ

Si x=y inicialmente y si la iteración de f dota a x de la propiedad φ el proceso puede repetirse para y.

Algunas proposiciones acerca del comportamiento de programas que se pueden expresar en DL son:

- Corrección parcial de α con respecto a φ , ψ : $\varphi \longrightarrow [\alpha]\psi$
- Corrección total débil de α con respecto a φ , ψ : $\varphi \longrightarrow \langle\alpha\rangle\psi$
- Corrección total fuerte de α con respecto a φ , ψ : $\varphi \longrightarrow ([\alpha]\psi \wedge \langle\alpha\rangle\psi)$
- El programa α es determinista

$\forall \bar{u}\ (\langle \alpha \rangle\ \bar{x}\doteq\bar{u} \longrightarrow [\alpha]\ \bar{x}\doteq\bar{u})$ con $\bar{x}$ todas las variables de α y $\bar{u}$ disjunto de $\bar{x}$.

- Los programas α y β son equivalentes (en cuanto a posibles resultados de cómputos convergentes)

$\forall \bar{u}\ (\langle\alpha\rangle\ \bar{x} \doteq \bar{u} \longleftrightarrow \langle\beta\rangle\ \bar{x} \doteq \bar{u})$ con $\bar{x}$ todas las varia-

bles de α y β y $\bar{u}$ disjunto de $\bar{x}$.

Observación: Para evitar problemas con la noción de variable libre en una fórmula dinámica φ vamos a adoptar una idea propuesta por Harel (18). Para un programa α que use variables $\bar{y}=y_1,.....,y_n$ y ψ una fórmula de DFS_d solo aceptamos $< \bar{y}:=\bar{x} ; \alpha ; \bar{x}:=\bar{y} > \psi$ tal que en ψ no aparecen $y_1,.....,y_n$ y en α no aparecen $x_1,.....,x_n$, como posibilidad de generar una nueva fórmula de DFS_d usando $< >$.

Es decir, restringimos $< \alpha > \psi$ de manera que las variables de entrada/salida $\bar{x}$ de α no se puedan utilizar en la ejecución de α y las variables de trabajo $\bar{y}$ no aparezcan en la fórmula ψ .

$\bar{y}:= \bar{x}$ es una abreviatura de $y_1:=x_1;.....;y_n:=x_n$

Para simplificar la notación seguiremos escribiendo $< \alpha > \psi$

Con esto, una aparición de una variable z en una fórmula ψ, $\psi \in DFS_d$ se llama ligada si aparece:

(i) dentro de una subfórmula de la forma $\exists z \psi_o$

(ii) dentro de una subfórmula de la forma $<..;z:=x;..;\alpha;..;x:=z;..>\psi_o$

(iii) dentro de una subfórmula de la forma $<..;y:=z;..;\alpha;..;z:=y;..>\psi_o$ considerando no ligada la aparición en $y:=x$

Toda aparición de una variable que no es ligada se llama libre. Para σ un término, x una variable y $\psi \in DFS_d$ obtenemos $\psi[\sigma/x]$ sustituyendo todas las apariciones libres de x en ψ por el término σ. Solo permitimos esta operación si ninguna variable de σ queda ligada en $\psi[\sigma/x]$.

Lema 1.3.1.- (Lema de sustitución)

Para cualquier $\mathfrak{D} \in STR_d$, $s \in W_{\mathfrak{D}}$, $\varphi \in DFS_d$, $\sigma \in TS_d$

$$\mathfrak{D} \models_s \varphi[\sigma/z] \iff \mathfrak{D} \models_{s[\sigma(s)\mathfrak{D}/z]} \varphi$$

Demostración:

Por inducción sobre la estructura de la fórmula φ

$\varphi \in FS_d$ es conocido.

$\varphi = \neg\psi$ Por hipótesis de inducción $\mathfrak{D} \models_s \psi[\sigma/z] \iff \mathfrak{D} \models_{s[\sigma(s)\mathfrak{D}/z]} \psi$

e inmediatamente se concluye $\mathfrak{D} \models_s \varphi[\sigma/z] \iff \mathfrak{D} \models_{s[\sigma(s)\mathfrak{D}/z]} \varphi$

$\varphi = \psi \wedge \chi$ Como $\varphi[\sigma/z] = \psi[\sigma/z] \wedge \chi[\sigma/z]$ aplicando la hipótesis de inducción obtenemos el resultado.

$\varphi = \exists x \psi$

Si $z = x$ $\varphi[\sigma/z] = \varphi$ y el resultado es inmediato

Si $z \neq x$ aplicamos la hipótesis de inducción a ψ y tenemos el lema

$\varphi = \langle \bar{y} := \bar{x}; \alpha; \bar{x} := \bar{y} \rangle \psi$ $\bar{y} = var(\alpha) = y_1, \ldots, y_n$ no aparecen en ψ, $\bar{x} = x_1, \ldots x_n$ no aparecen en α

Si z no está en $\bar{x}$ ni en $\bar{y}$ $\varphi[\sigma/z] = \langle \bar{y} := \bar{x}; \alpha; \bar{x} := \bar{y} \rangle \psi[\sigma/z]$ y por hipótesis de inducción $\mathfrak{D} \models_s \psi[\sigma/z] \iff \mathfrak{D} \models_{s[\sigma(s)\mathfrak{D}/z]} \psi$

$\mathfrak{D} \models_s \langle \bar{y} := \bar{x}; \alpha; \bar{x} := \bar{y} \rangle \psi[\sigma/z]$ $\iff$ existe $t \in W_{\mathfrak{D}}$ $s(\bar{y} := \bar{x}; \alpha; \bar{x} := \bar{y})^{\mathfrak{D}} t$ y $\mathfrak{D} \models_t \psi[\sigma/z]$ $\iff$ (z no aparece en $\bar{x}$ ni en $\bar{y}$ e hipótesis de inducción) $s[\sigma(s)^{\mathfrak{D}}/z](\bar{y} := \bar{x}; \alpha; \bar{x} := \bar{y})^{\mathfrak{D}} t[\sigma(s)^{\mathfrak{D}}/z]$ y $\mathfrak{D} \models_{t[\sigma(s)\mathfrak{D}/z]} \psi$ $\iff$ $\mathfrak{D} \models_{s[\sigma(s)\mathfrak{D}/z]} \varphi$

Si $z = y_i$ $\varphi[\sigma/z] = \varphi$ y el resultado es inmediato pues z no apa-

rece en φ

Si $z \in \bar{x}$ podemos suponer $z = x_1$

$\varphi[\sigma/z] = \langle y_1 := \sigma ; y_2 := x_2 ; .. ; y_n := x_n ; \alpha ; \bar{x} := \bar{y} \rangle \varphi$

$\mathfrak{D} \models_s \varphi[\sigma/z] \iff$ existe $t \in W_{\mathfrak{D}}$ $s(y_1 := \sigma ; .. ; y_n := x_n ; \alpha ; \bar{x} := \bar{y})^{\mathfrak{D}} t$

y $\mathfrak{D} \models_t \psi \iff s[\sigma(s)^{\mathfrak{D}}/y_1](y_2 := x_2 ; .. ; y_n := x_n ; \alpha ; \bar{x} := \bar{y})^{\mathfrak{D}} t$ y

$\mathfrak{D} \models_s \psi \iff s[\sigma(s)^{\mathfrak{D}}/z](y_1 := x_1 ; .. ; y_n := x_n ; \alpha ; \bar{x} := \bar{y})^{\mathfrak{D}} t$ y $\mathfrak{D} \models_t \psi$

$\iff \mathfrak{D} \models_{s[\sigma(s)^{\mathfrak{D}}/z]} \varphi$

Lema 1.3.2.-

Para cualquier $\alpha, \beta \in PS_d$, $\varphi, \chi \in DFS_d$

(a) $\models (\langle x := \tau \rangle \psi \leftrightarrow \psi[\tau/x])$

(b) $\models (\langle \chi ? \rangle \psi \leftrightarrow (\chi \wedge \psi))$

(c) $\models (\langle \alpha ; \beta \rangle \psi \leftrightarrow \langle \alpha \rangle \langle \beta \rangle \psi)$

(d) $\models (\langle \alpha \cup \beta \rangle \psi \leftrightarrow \langle \alpha \rangle \psi \vee \langle \beta \rangle \psi)$

Demostración:

Sea $s \ W_d$

(a) $\mathfrak{D} \models_s \langle x := \tau \rangle \psi \iff$ existe $t \in W_{\mathfrak{D}}$ $s(x := \tau)^{\mathfrak{D}} t$ y $\mathfrak{D} \models_t \psi$

$\iff t = s[\tau(s)^{\mathfrak{D}}/x]$ y $\mathfrak{D} \models_t \psi$

$\iff \mathfrak{D} \models_{s[\tau(s)^{\mathfrak{D}}/x]} \psi$

$\iff$ por el lema 1.3.1 $\mathfrak{D} \models_s \psi[\tau/x]$

(b) $\mathfrak{D} \models_s \langle \chi ? \rangle \psi \iff$ existe $t \in W_{\mathfrak{D}}$ $s(\chi ?)^{\mathfrak{D}} t$ y $\mathfrak{D} \models_t \psi$

$\iff s = t, \mathfrak{D} \models_s \chi$ y $\mathfrak{D} \models_t \psi$

$\iff \mathfrak{D} \models_s \chi \wedge \psi$

(c) $\mathfrak{D} \models_s \langle \alpha ; \beta \rangle \psi$ $\Longleftrightarrow$ existe $t \in W_{\mathfrak{D}}$ $s(\alpha ; \beta)^{\mathfrak{D}} t$ y $\mathfrak{D} \models_t \psi$

$\Longleftrightarrow$ existen $t, r \in W_{\mathfrak{D}}$ $s\, \alpha^{\mathfrak{D}} r\, \beta^{\mathfrak{D}} t$ y $\mathfrak{D} \models_t \psi$

$\Longleftrightarrow$ existe $r \in W_{\mathfrak{D}}$ $s\, \alpha^{\mathfrak{D}} r$ y $\mathfrak{D} \models_r \langle \beta \rangle \psi$

$\Longleftrightarrow$ $\mathfrak{D} \models_s \langle \alpha \rangle \langle \beta \rangle \psi$

(d) $\mathfrak{D} \models_s \langle \alpha \cup \beta \rangle \psi$ $\Longleftrightarrow$ existe $t \in W_{\mathfrak{D}}$ $s(\alpha \cup \beta)^{\mathfrak{D}} t$ y $\mathfrak{D} \models_t \psi$

$\Longleftrightarrow$ existe $t \in W_{\mathfrak{D}}$ ($s\, \alpha^{\mathfrak{D}} t$ y $\mathfrak{D} \models_t \psi$ o $s \beta^{\mathfrak{D}} t$ y $\mathfrak{D} \models_t \psi$)

$\Longleftrightarrow$ $\mathfrak{D} \models_s \langle \alpha \rangle \psi$ o $\mathfrak{D} \models_s \langle \beta \rangle \psi$

$\Longleftrightarrow$ $\mathfrak{D} \models_s \langle \alpha \rangle \psi \vee \langle \beta \rangle \psi$

1.4.- Axiomatización infinitaria para DL. El cálculo IDL. Teorema de completitud.

Un cálculo completo para DL fue propuesto por Mirkowska (31) y estructurado por Harel (18). Se basa en las ideas de los cálculos para PDL y la lógica de primer orden, con una regla infinitaria (ω).

Definición 1.4.1.- Cálculo infinitario IDL para DL

Axiomas

(T∃) Todas las instancias de tautologías del cálculo proposicional y axiomas para el ∃ y la igualdad, resultantes de sustituir consistentemente variables proposicionales o fórmulas por fórmulas de DFS_d.

(:=) $\langle x := \tau \rangle \varphi \leftrightarrow \varphi[\tau/x]$

(?) $\langle X? \rangle \varphi \leftrightarrow X \wedge \varphi$

(U) $\langle \alpha \cup \beta \rangle \varphi \leftrightarrow \langle \alpha \rangle \varphi \vee \langle \beta \rangle \varphi$

(;) $\langle \alpha ; \beta \rangle \varphi \leftrightarrow \langle \alpha \rangle \langle \beta \rangle \varphi$

(*) $\langle \alpha^* \rangle \varphi \leftrightarrow (\varphi \vee \langle \alpha \rangle \langle \alpha^* \rangle \varphi)$

Reglas

(MP) $$\frac{\varphi \ , \ \varphi \rightarrow \psi}{\psi}$$

(I∃) $$\frac{\varphi[w/x] \rightarrow \psi}{\exists x \varphi \rightarrow \psi}$$ w variable o constante

(G1) $$\frac{\varphi \rightarrow \psi}{\langle \alpha \rangle \varphi \rightarrow \langle \alpha \rangle \psi}$$

(G2) $$\frac{\varphi \longrightarrow \psi}{\exists x \varphi \longrightarrow \exists x \psi}$$

(∞) $$\frac{\{ \varphi \longrightarrow [\alpha^n]\psi \} \quad n \in N}{\varphi \longrightarrow [\alpha^*]\psi}$$

En ($T\exists$) hay axiomas como las leyes de Morgan, axiomas para cuantificadores como $\varphi[\sigma/x] \longrightarrow \exists x \varphi$, axiomas para $\doteq$ como $x \doteq x$ $x \doteq y \wedge \varphi \longrightarrow \varphi[y/x]$ (φ atómica o negación de atómica)

La aplicación de la regla ($I\exists$) solo esta permitida si se cumple una condición crítica: la premisa de la regla debe haberse derivado de un conjunto $\tilde{\Phi}$ tal que w no aparezca (o no aparezca libre si es una variable) en $\tilde{\Phi}$ ni en la conclusión de la regla.

Definición 1.4.2.- Derivabilidad en IDL : $\vdash^{IDL}$

$\tilde{\Phi} \vdash^{IDL} \varphi$ significa que existe una demostración formal de φ a partir de $\tilde{\Phi}$ en $\vdash^{IDL}$. Esta es un árbol con fórmulas en los nodos que cumple:

(a) φ está en la raiz

(b) la fórmula de cada nodo γ es un axioma de IDL o una hipótesis tomada de $\tilde{\Phi}$, o el resultado de aplicar una regla de IDL al conjunto de fórmulas de los hijos del nodo, tomadas como premisas.

(c) si un nodo γ con fórmula $\exists x \varphi \longrightarrow \psi$ resulta de aplicar ($I\exists$) a la fórmula $\varphi[w/x] \longrightarrow \psi$ del único hijo γ' de γ , entonces w no aparece (no aparece libre si es una variable) en $\exists x \varphi \longrightarrow \psi$ ni en nin-

guna de las fórmulas de γ' o de los descendientes de γ' que hayan sido introducidas en la demostración como hipótesis de Φ .

Teorema 1.4.1.- Corrección de $\vdash^{IDL}$

Para cualquier $\Phi \subseteq DFS_d$, $\varphi \in DFS_d$

$$\Phi \vdash^{IDL} \varphi \implies \Phi \models \varphi$$

Demostración:

Basta comprobar que los axiomas son válidos en cualquier estructura $\mathfrak{D}$ y para las reglas si las premisas son válidas en $\mathfrak{D}$ ($\mathfrak{D} \models \Phi$) lo es también la conclusión.

La corrección de (:=), (?), (;) y ($\cup$) se pueden encontrar en el lema 1.2.2.

Corrección de (*)

$$\mathfrak{D} \models (\varphi \vee \langle\alpha\rangle\langle\alpha^*\rangle\varphi) \leftrightarrow \langle\alpha^*\rangle\varphi$$

$\leftarrow$) Sea $s \in W_{\mathfrak{D}}$

$\mathfrak{D} \models_s \langle\alpha^*\rangle\varphi \iff$ existen $s=r_0,\ldots,r_n \in W_{\mathfrak{D}}$ tal que $r_0 \alpha^{\mathfrak{D}} r_1 \ldots \alpha^{\mathfrak{D}} r_n$ y $\mathfrak{D} \models_{r_n} \varphi$

Si $n=0 \implies \mathfrak{D} \models_s \varphi$

Si $n>0 \implies \mathfrak{D} \models_{r_1} \langle\alpha^*\rangle\varphi$ y $s\, \alpha^{\mathfrak{D}} r_1 \implies \mathfrak{D} \models_s \langle\alpha\rangle\langle\alpha^*\rangle\varphi$

Luego $\mathfrak{D} \models_s \varphi \vee \langle\alpha\rangle\langle\alpha^*\rangle\varphi$

$\rightarrow$) Sea $s \in W_{\mathfrak{D}}$

Si $\mathfrak{D} \models_s \varphi \implies$ existe $r_0 = s$ tal que $\mathfrak{D} \models_s \varphi \implies \mathfrak{D} \models_s \langle\alpha^*\rangle\varphi$

Si $\mathfrak{D} \models_s \langle\alpha\rangle\langle\alpha^*\rangle\varphi \implies$ existe $r_1 \in W_{\mathfrak{D}}$ tal que $s\, \alpha^{\mathfrak{D}} r_1$ y $\mathfrak{D} \models_{r_1} \langle\alpha^*\rangle\varphi$

$\Longrightarrow$ existen $r_1,\ldots,r_n$ con $s\ \alpha^{\mathfrak{D}}\ r_1\ \alpha^{\mathfrak{D}}\ \ldots\ \alpha^{\mathfrak{D}}\ r_n$ y $\mathfrak{D} \models_{r_n} \varphi$ $\Longrightarrow$

$\mathfrak{D} \models_s \langle \alpha^* \rangle \varphi$

Corrección de (G1)

Sean φ, ψ fórmulas de DFS_d y $\alpha \in PS_d$ y $\mathfrak{D}$ estructura con

$\mathfrak{D} \models \varphi \longrightarrow \psi$ entonces $\mathfrak{D} \models \langle \alpha \rangle \varphi \longrightarrow \langle \alpha \rangle \psi$

Sea $s \in W_{\mathfrak{D}}$

$\mathfrak{D} \models_s \langle \alpha \rangle \varphi \Longrightarrow$ existe $t \in W_{\mathfrak{D}}$ con $s\ \alpha^{\mathfrak{D}}\ t$ y $\mathfrak{D} \models_t \varphi$

Como $\mathfrak{D} \models \varphi \longrightarrow \psi$ tenemos $\mathfrak{D} \models_t \psi \Longrightarrow \mathfrak{D} \models_s \langle \alpha \rangle \psi$

Corrección de (∞)

Si para todo $n \in N$ $\mathfrak{D} \models \varphi \longrightarrow [\alpha^n]\psi \Longrightarrow \mathfrak{D} \models \varphi \longrightarrow [\alpha^*]\psi$

Sea $s \in W_{\mathfrak{D}}$; para todo $n \in N$ se tendrá

$\mathfrak{D} \models_s \varphi \longrightarrow [\alpha^n]\psi$ y supongamos $\mathfrak{D} \models_s \varphi$

Sea $t \in W_{\mathfrak{D}}$ con $s\ \alpha^{\mathfrak{D}}\ t \Longrightarrow$ existen $r_o = s,\ldots\ldots,r_n = t \in W_{\mathfrak{D}}$ tal que

$r_o\ \alpha^{\mathfrak{D}}\ r_1\ \alpha^{\mathfrak{D}}\ \ldots\ \alpha^{\mathfrak{D}}\ r_n \Longrightarrow s\ \alpha^{n\,\mathfrak{D}}\ t$

Como $\mathfrak{D} \models_s \varphi \Longrightarrow \mathfrak{D} \models_s [\alpha^n]\psi \Longrightarrow (s\ \alpha^{n\,\mathfrak{D}}\ t)\ \mathfrak{D} \models_t \psi \Longrightarrow$

$\mathfrak{D} \models_s [\alpha^*]\psi$

Corrección de (I$\exists$)

La corrección de (I$\exists$) se enuncia:

w no aparece (no aparece libre) en Φ ni en $\exists x \varphi \longrightarrow \psi$

$\Phi \models \varphi[w/x] \longrightarrow \psi \Longrightarrow \Phi \models \exists x\ \varphi \longrightarrow \psi$

y se comprueba usando el lema de sustitución.

El resto de axiomas y reglas son claramente válidos.

Lema 1.4.1.- **Axiomas y reglas duales del cálculo IDL**

Axiomas duales de IDL

(:='') $[x:=\tau]\varphi \leftrightarrow \varphi[\tau/x]$

(?') $[\chi?]\varphi \leftrightarrow (\chi \rightarrow \varphi)$

(∪') $[\alpha \cup \beta]\varphi \leftrightarrow ([\alpha]\varphi \wedge [\beta]\varphi)$

(;') $[\alpha ; \beta]\varphi \leftrightarrow [\alpha][\beta]\varphi$

(*') $[\alpha^*]\varphi \rightarrow [\alpha^n]\varphi$ **para cualquier** $n \in N$

Reglas duales de IDL

(G1') $$\frac{\varphi \rightarrow \psi}{[\alpha]\varphi \rightarrow [\alpha]\psi}$$

(G2') $$\frac{\varphi \rightarrow \psi}{\forall x\, \varphi \rightarrow \forall x\, \psi}$$

(I∀) $$\frac{\varphi \rightarrow \psi[w/x]}{\varphi \rightarrow \forall x\, \psi}$$

(∞') $$\frac{\langle\alpha^n\rangle\varphi \rightarrow \psi \quad n \in N}{\langle\alpha^*\rangle\varphi \rightarrow \psi}$$

Demostración: Derivacion de estas reglas en IDL:

(:='')

(:=) $\vdash^{IDL} \langle x:=\tau\rangle\neg\varphi \leftrightarrow \neg\varphi[\tau/x]$

(T∃) $\vdash^{IDL} \neg\neg\varphi[\tau/x] \leftrightarrow \neg\langle x:=\tau\rangle\neg\varphi$

(T∃) $\vdash^{IDL} \varphi[\tau/x] \leftrightarrow [x:=\tau]\varphi$

(?''), (∪''), (;'') **se demuestran análogamente.**

(G1')

$$\vdash^{IDL} \varphi \longrightarrow \psi$$

(T∃) $\vdash^{IDL} \neg\psi \longrightarrow \neg\varphi$

(G1) $\vdash^{IDL} \langle\alpha\rangle\neg\psi \longrightarrow \langle\alpha\rangle\neg\varphi$

(T∃) $\vdash^{IDL} \neg\langle\alpha\rangle\neg\varphi \longrightarrow \neg\langle\alpha\rangle\neg\psi$

$\vdash^{IDL} [\alpha]\varphi \longrightarrow [\alpha]\psi$

(G2'), (I∀), (∞') se demuestran análogamente. Para aplicar (I∀) se necesita la misma condición vista para (I∃)

($_*$')

Análogamente a lo anterior se puede probar

(1) $\vdash^{IDL} [\alpha^*]\varphi \longleftrightarrow \varphi \wedge [\alpha][\alpha^*]\varphi$

Por inducción sobre n

n=0 $[\alpha^0] = \varphi$

(T∃) $\vdash^{IDL} [\alpha^*]\varphi \longleftrightarrow \varphi \wedge [\alpha][\alpha^*]\varphi$ (1)

(T∃) $\vdash^{IDL} [\alpha^*]\varphi \longrightarrow \varphi$

n+1 supuesto para n

Por hipótesis $\vdash^{IDL} [\alpha^*]\varphi \longrightarrow [\alpha^n]\varphi$

(G1') $\vdash^{IDL} [\alpha][\alpha^*]\varphi \longrightarrow [\alpha][\alpha^n]\varphi$

(T∃) $\vdash^{IDL} (\varphi \wedge [\alpha][\alpha^*]\varphi) \longrightarrow [\alpha][\alpha^n]\varphi$

(1), (MP) $\vdash^{IDL} [\alpha^*]\varphi \longrightarrow [\alpha][\alpha^n]\varphi$

(T∃),(MP),(;') $\vdash^{IDL} [\alpha^*]\varphi \longrightarrow [\alpha^{n+1}]\varphi$

El cálculo $\vdash^{IDL}$ es completo para DL. Para probarlo traduciremos DL a un fragmento de la lógica $(L_{\omega_1\omega})_d$ y aplicaremos el teorema de existencia de modelos (cfr. (26)).

Para simplificar llamaremos $L_{\omega_1\omega}$ a $(L_{\omega_1\omega})_d$ (que es la lógica infinitaria $L_{\omega_1\omega}$ para los símbolos de d).

En este desarrollo no conviene tratar las abreviaturas dadas ($\forall$, $\vee$, $[\,]$) como negaciones, sino como casos distinguidos. Asi lo haremos.

Definición 1.4.3.- Complejidad de programas y fórmulas de DL

Definimos una función de complejidad de programas y fórmulas: c que toma ordinales como valores.

$c(x := \tau) = 1$

$c(\alpha ; \beta) = c(\alpha) + c(\beta) + 2$

$c(\alpha \cup \beta) = \max\{c(\alpha), c(\beta)\} + 1$

$c(\alpha^*) = c(\alpha) + \omega$

$c(\varphi?) = c(\varphi) + 1$

$c(\varphi) = 0$ si φ de primer orden libre de cuantificadores

$c(\varphi \vee \psi) = c(\varphi \wedge \psi) = \max\{c(\varphi), c(\psi)\} + 1$

$c(\forall x \varphi) = c(\exists x \varphi) = c(\varphi) + 1$

$c([\alpha]\varphi) = c(\langle\alpha\rangle\varphi) = c(\alpha) + c(\varphi) + 1$

$c(\neg\varphi) = c(\varphi) + 1$ salvo en los casos anteriores.

Lema 1.4.2.-

Existe una traducción ν de fórmulas de DL a fórmulas de $L_{\omega_1\omega}$ tal que para cualquier estructura $\mathcal{D}$ y estado $s \in W_{\mathcal{D}}$

$$\mathcal{D} \models_s \varphi \iff \mathcal{D} \models_s \nu(\varphi)$$

Demostración:

Se define la traducción

$\nu(\varphi) = \varphi$ si φ es de primer orden

$\nu(\varphi \wedge \psi) = \nu(\varphi) \wedge \nu(\psi)$

$\nu(\varphi \vee \psi) = \nu(\varphi) \vee \nu(\psi)$

$\nu(\exists x\, \varphi) = \exists x\, \nu(\varphi)$

$\nu(\forall x\, \varphi) = \forall x\, \nu(\varphi)$

$\nu(\langle x := \tau \rangle \varphi) = \nu(\varphi[\tau/x])$

$\nu([x := \tau]\varphi) = \nu(\varphi[\tau/x])$

$\nu(\langle \alpha ; \beta \rangle \varphi) = \nu(\langle \alpha \rangle \langle \beta \rangle \varphi)$

$\nu([\alpha ; \beta]\varphi) = \nu([\alpha][\beta]\varphi)$

$\nu(\langle \alpha \cup \beta \rangle \varphi) = \nu(\langle \alpha \rangle \varphi) \vee \nu(\langle \beta \rangle \varphi)$

$\nu([\alpha \cup \beta]\varphi) = \nu([\alpha]\varphi) \wedge \nu([\beta]\varphi)$

$\nu(\langle \chi? \rangle) = \nu(\chi) \wedge \nu(\varphi)$

$\nu([\chi?]\varphi) = \nu(\chi) \longrightarrow \nu(\varphi)$

$\nu(\langle \alpha^* \rangle \varphi) = \bigvee_{n \geq 0} \nu(\langle \alpha^n \rangle \varphi)$

$\nu([\alpha^*]\varphi) = \bigwedge_{n \geq 0} \nu([\alpha^n]\varphi)$

$\nu(\neg \varphi) = \neg \nu(\varphi)$ salvo en los casos anteriores.

La demostración del lema es ahora inmediata por inducción sobre la complejidad $c(\varphi)$

La traducción ν no se comporta todo lo bien que fuera deseable. Por ejemplo, $\nu(\varphi)$ puede ser de la forma $\psi_1 \vee \psi_2$, mientras φ no es una disyunción (si φ es $\langle x:=\tau\rangle(\varphi_1 \vee \varphi_2)$ o $\langle\alpha \cup \beta\rangle\varphi_0$). Esto complica las demostraciones por inducción sobre la complejidad.

Para evitar esto se utiliza una técnica propuesta por Keisler en (26), definiendo $\varphi\neg$.

$\varphi\neg$ es una fórmula equivalente a $\neg\varphi$ pero introduciendo la negación hasta que solo afecte a fórmulas atómicas y eliminando las apariciones de negaciones consecutivas.

Definición 1.4.4.-

Para $\varphi \in DFS_d$ o $\varphi \in (L\omega_1\omega)_d$

$\varphi\neg = \neg\varphi$ si φ es atómica

$(\varphi \wedge \psi)\neg = \varphi\neg \vee \psi\neg$

$(\varphi \vee \psi)\neg = \varphi\neg \wedge \psi\neg$

$(\exists x \varphi)\neg = \forall x \varphi\neg$

$(\forall x \varphi)\neg = \exists x \varphi\neg$

$(\langle\alpha\rangle\varphi)\neg = [\alpha]\varphi\neg$

$([\alpha]\varphi)\neg = \langle\alpha\rangle\varphi\neg$

$(\bigwedge_{n\geq 0}\varphi_n)\neg = \bigvee_{n\geq 0}\varphi_n\neg$

$(\bigvee_{n\geq 0}\varphi_n)\neg = \bigwedge_{n\geq 0}\varphi_n\neg$

$(\neg\varphi)\neg = \varphi$ salvo en los casos anteriores

Lema 1.4.3

Para cualquier $\varphi \in DFS_d$ $\quad \vdash^{IDL} \neg\varphi \leftrightarrow \varphi\neg$

Demostración:

Inmediata usando (T∃), (G1), (G2) y las definiciones de $\vee$, $\forall$ y $[\ ]$.

Lema 1.4.4.-

Para cualquier fórmula $\varphi \in DFS_d$

(a) $\nu(\varphi\neg) = \nu(\varphi)\neg$

(b) $\nu(\varphi\,[\tau/x]) = \nu(\varphi)\,[\tau/x]$

Demostración:

Inmediata de las definiciones de ν y $\varphi\neg$, por inducción sobre la complejidad $c(\varphi)$.

Notaciones:

C denota un conjunto infínito numerable de constantes que no aparecen en d

$\bar{d} = d \cup C$

$L_{\omega_1\omega} = (L_{\omega_1\omega})_d$

$L_{\omega_1\omega}(C) = (L_{\omega_1\omega})_{\bar{d}}$

$DFS(C) = DFS_{\bar{d}}$

$Con^{IDL}(\Phi)$ indica que $\Phi \vdash^{IDL} \not$ **false**

$\Phi \vdash_C^{IDL} \varphi$ (con $\Phi \subseteq DFS(C)$, $\varphi \in DFS(C)$) y $Con_C^{IDL}(\Phi)$ tienen significados análogos permitiendo símbolos de $\bar{d}$ (ver definición 1.4.2)

Lema 1.4.5.-

Para cualquier $\varphi \in DFS(C)$ se verifica

(i) $\nu(\varphi) = \chi$ atómica o atómica negada $\Longrightarrow \vdash^{IDL}_{C} \chi \leftrightarrow \varphi$

(ii) $\nu(\varphi) = \neg\chi \Longrightarrow$ existe $\psi \in DFS(C)$ tal que

$$\nu(\psi) = \chi\neg \quad y \quad \vdash^{IDL}_{C} \psi \leftrightarrow \varphi$$

(iii) $\nu(\varphi) = \chi_1 \wedge \chi_2 \Longrightarrow$ existen $\varphi_1, \varphi_2 \in DFS(C)$ tal que

$$\nu(\varphi_1) = \chi_1, \ \nu(\varphi_2) = \chi_2 \ y \ \vdash^{IDL}_{C} (\varphi_1 \wedge \varphi_2) \leftrightarrow \varphi$$

(iv) $\nu(\varphi) = \chi_1 \vee \chi_2 \Longrightarrow$ existen $\varphi_1, \varphi_2 \in DFS(C)$ tal que

$$\nu(\varphi_1) = \chi_1, \ \nu(\varphi_2) = \chi_2 \ y \ \vdash^{IDL}_{C} (\varphi_1 \vee \varphi_2) \leftrightarrow \varphi$$

(v) $\nu(\varphi) = \bigwedge_{n>0} \chi_n \Longrightarrow$ existe $[\alpha^*]\psi \in DFS(C)$ tal que

$\nu([\alpha^n]\psi) = \chi_n$ para cualquier $n \in N$ y $\vdash^{IDL}_{C} [\alpha^*]\psi \leftrightarrow \varphi$

(vi) $\nu(\varphi) = \bigvee_{n>0} \chi_n \Longrightarrow$ existe $\langle\alpha^*\rangle\psi \in DFS(C)$ tal que

$\nu(\langle\alpha^n\rangle\psi) = \chi_n$ para cualquier $n \in N$ y $\vdash^{IDL}_{C} \langle\alpha^*\rangle\psi \leftrightarrow \varphi$

(vii) $\nu(\varphi) = \exists x \chi \Longrightarrow$ existe $\exists x \psi \in DFS(C)$ tal que

$$\nu(\psi) = \chi \quad y \quad \vdash^{IDL}_{C} \exists x \psi \leftrightarrow \varphi$$

(viii) $\nu(\varphi) = \forall x \chi \Longrightarrow$ existe $\forall x \psi \in DFS(C)$ tal que

$$\nu(\psi) = \chi \quad y \quad \vdash^{IDL}_{C} \forall x \psi \leftrightarrow \varphi$$

Demostración:

Se procede en las ocho partes por inducción sobre $c(\varphi)$

(i) $\nu(\varphi) = \chi$ atómica o atómica negada

$c(\varphi) = 0 \Longrightarrow \varphi$ es de primer orden y $\nu(\varphi) = \varphi = \chi$

Con (T$\exists$) $\quad \vdash^{IDL}_{C} \varphi \leftrightarrow \chi$

$c(\varphi) > 0$

- $\varphi = \psi_1 \wedge \psi_2$

$\nu(\varphi) = \nu(\psi_1) \wedge \nu(\psi_2)$ que no es atómica ni atómica negada, luego este caso no es posible.

- $\varphi = \psi_1 \vee \psi_2$ este caso tampoco es posible.

- $\varphi = \forall x \psi_o$
- $\varphi = \exists x \psi_o$

} casos no posibles

- $\varphi = \neg \psi_o$

$\nu(\varphi) = \nu(\neg\psi_o) = \neg \nu(\psi_o) = \chi$ y χ debe ser una fórmula atómica negada $\chi = \neg \chi_o$ con χ_o atómica y $\nu(\psi_o) = \chi_o$

Además $c(\varphi) = c(\psi_o) + 1 \Longrightarrow c(\psi_o) < c(\varphi)$, luego podemos aplicar la hipótesis de inducción y $\vdash^{IDL}_{C} \psi_o \leftrightarrow \chi_o$ y

$$(T\exists)\ \vdash^{IDL}_{C} \neg\chi_o \leftrightarrow \neg\psi_o$$
$$\vdash^{IDL}_{C} \chi \leftrightarrow \varphi$$

- $\varphi = \langle x := \tau \rangle \psi_o$

$\nu(\varphi) = \nu(\psi_o\ [\tau/x]) = \chi$

$c(\varphi) = c(\psi_o) + 2 < c(\psi_o) = c(\psi_o\ [\tau/x])$

Podemos aplicar la hipótesis de inducción y

$$\vdash^{IDL}_{C} \psi_o\ [\tau/x] \leftrightarrow \chi$$
$$(:=),(T\exists),(MP)\quad \vdash^{IDL}_{C} \langle x := \tau \rangle \psi_o \leftrightarrow \chi$$
$$\vdash^{IDL}_{C} \varphi \leftrightarrow \chi$$

- $\varphi = [x := \tau] \psi_o$ análogo al anterior usando $(:=')$

• $\varphi = \langle \alpha ; \beta \rangle \psi_0$

$\nu(\varphi) = \nu(\langle \alpha \rangle \langle \beta \rangle \psi_0) = \chi$

$c(\langle \alpha \rangle \langle \beta \rangle \psi_0) = c(\alpha) + c(\beta) + c(\psi_0) + 2 < c(\langle \alpha ; \beta \rangle \psi_0) =$

$c(\alpha) + c(\beta) + c(\psi_0) + 3$

Aplicamos la hipótesis de inducción a $\langle \alpha \rangle \langle \beta \rangle \psi_0$

$$\vdash_{C}^{IDL} \langle \alpha \rangle \langle \beta \rangle \psi_0 \leftrightarrow \chi$$

$(;),(T\exists),(MP) \vdash_{C}^{IDL} \langle \alpha ; \beta \rangle \psi_0 \leftrightarrow \chi$

• $\varphi = \langle \alpha \cup \beta \rangle \psi_0$

$\nu(\varphi) = \nu(\langle \alpha \rangle \psi_0) \vee \nu(\langle \beta \rangle \psi_0)$ que no es atómica ni atómica negada, luego este caso no es posible.

• $\varphi = [\alpha ; \beta] \psi_0$ es análogo al caso con $\langle \ \rangle$ usando $(;')$

• $\varphi = [\alpha \cup \beta] \psi_0$ caso no posible

• $\varphi = \langle \psi_1 ? \rangle \psi_2$

$\nu(\varphi) = \nu(\psi_1) \wedge \nu(\psi_2)$ que no es atómica ni atómica negada, luego este caso no es posible.

• $\varphi = [\psi_1 ?] \psi_2$ caso no posible

• $\varphi = \langle \alpha^* \rangle \psi_0$

$\nu(\varphi) = \bigvee_{n \geq 0} \nu(\langle \alpha^n \rangle \psi_0)$ que no es atómica ni atómica negada, luego este caso no es posible.

• $\varphi = [\alpha^*] \psi_0$ caso no posible

(ii) $\nu(\varphi) = \neg\chi$

$c(\varphi) = 0 \implies \varphi$ de primer orden sin cuantificadores.

$\nu(\varphi) = \varphi = \neg\chi$, luego χ también es de primer orden sin cuantificadores. Tomamos $\psi = \chi\neg$ y resulta $\nu(\psi) = \chi\neg$

y $\vdash^{IDL}_{C}$ $\psi \leftrightarrow \varphi$ porque $\vdash^{IDL}_{C}$ $\chi\neg \leftrightarrow \chi$ por el lema 1.4.3

$c(\psi) > 0$

Los únicos casos posibles son:

- $\varphi = \neg\psi_0$

$$\nu(\varphi) = \neg\nu(\psi_0) = \neg\chi \implies \nu(\psi_0) = \chi$$

Tomamos $\psi = \psi_0\neg$ y $\nu(\psi) = \nu(\psi_0\neg) = \nu(\psi_0)\neg = \chi\neg$ (Lema 1.4.4)

Por el lema 1.4.3 $\vdash^{IDL}_{C}$ $\psi \leftrightarrow \varphi$

- $\varphi = \langle x := \tau\rangle\psi_0$
- $\varphi = [x := \tau]\psi_0$
- $\varphi = \langle\alpha;\beta\rangle\psi_0$
- $\varphi = [\alpha;\beta]\psi_0$

} análogo a (i)

- $\varphi = \langle\alpha \cup \beta\rangle\psi_0$

$\nu(\varphi) = \nu(\langle\alpha\rangle\psi_0) \vee \nu(\langle\beta\rangle\psi_0) = \neg(\neg\nu(\langle\alpha\rangle\psi_0) \wedge \neg\nu(\langle\beta\rangle\psi_0))$

$= \neg\chi \implies \chi = \neg\nu(\langle\alpha\rangle\psi_0) \wedge \neg\nu(\langle\beta\rangle\psi_0)$

Tomamos $\psi = \langle\alpha\rangle\psi_0 \vee \langle\beta\rangle\psi_0$ y

$$\nu(\psi) = \nu(\langle\alpha\rangle\psi_0) \vee \nu(\langle\beta\rangle\psi_0) = \chi\neg$$

y por (U) $\vdash^{IDL}_{C}$ $\psi \leftrightarrow \varphi$

- $\varphi = \langle\alpha^*\rangle\psi_0$

$$\nu(\varphi) = \bigvee_{n\geq 0} \nu(\langle\alpha^n\rangle\psi_0) = \neg\bigwedge_{n\geq 0} \neg\nu(\langle\alpha^n\rangle\psi_0) = \neg\chi$$

$\Longrightarrow \quad \chi = \bigwedge_{n\geq 0} \neg\nu(\langle\alpha^n\rangle\psi_o)$ Tomamos $\psi = \varphi$ y

$\nu(\psi) = \bigvee_{n\geq 0} \nu(\langle\alpha^n\rangle\psi_o) = \chi\neg$ y

$$(T\exists) \quad \vdash^{IDL}_{C} (\psi \leftrightarrow \varphi)$$

(iii) $\nu(\varphi) = \chi_1 \wedge \chi_2$

$c(\varphi) = 0 \Longrightarrow$ φ de primer orden libre de cuantificadores

$\nu(\varphi) = \varphi = \chi_1 \wedge \chi_2 \Longrightarrow \varphi = \varphi_1 \wedge \varphi_2$ con φ_1, φ_2 de primer orden sin cuantificadores

$$\nu(\varphi_1) = \chi_1 \qquad \nu(\varphi_2) = \chi_2$$

y $(T\exists) \vdash^{IDL}_{C} (\varphi_1 \wedge \varphi_2) \leftrightarrow \varphi$

$c(\varphi) > 0$

Los únicos casos posibles son:

- $\varphi = \psi_1 \wedge \psi_2$

$$\nu(\varphi) = \nu(\psi_1) \wedge \nu(\psi_2) = \chi_1 \wedge \chi_2 \Longrightarrow \begin{matrix} \nu(\psi_1) = \chi_1 \\ \nu(\psi_2) = \chi_2 \end{matrix}$$

Tomando $\begin{matrix} \varphi_1 = \psi_1 \\ \varphi_2 = \psi_2 \end{matrix}$ $(T\exists) \vdash^{IDL}_{C} (\varphi_1 \wedge \varphi_2) \leftrightarrow \varphi$

- $\varphi = \langle x := \tau\rangle\psi_o$
- $\varphi = [x := \tau]\psi_o$
- $\varphi = \langle\alpha;\beta\rangle\psi_o$
- $\varphi = [\alpha;\beta]\psi_o$

} análogos a (i)

- $\varphi = [\alpha \cup \beta]\psi_o$

$\nu(\varphi) = \nu([\alpha]\psi_o) \wedge \nu([\beta]\psi_o) = \chi_1 \wedge \chi_2 \Longrightarrow$

$\chi_1 = \nu([\alpha]\psi_0)$ $\qquad$ $\varphi_1 = [\alpha]\psi_0$

Tomando

$\chi_2 = \nu([\beta]\psi_0)$ $\qquad$ $\varphi_2 = [\beta]\psi_0$

$$(\cup') \quad \vdash^{IDL}_{C} (\varphi_1 \wedge \varphi_2) \leftrightarrow \varphi$$

- $\varphi = \langle \psi_1 ? \rangle \psi_2$

$$\nu(\psi) = \nu(\psi_1) \wedge \nu(\psi_2) = \chi_1 \wedge \chi_2 \Longrightarrow$$

$\chi_1 = \nu(\psi_1)$ $\qquad$ $\varphi_1 = \psi_1$

Tomando $\qquad$ tenemos

$\chi_2 = \nu(\psi_2)$ $\qquad$ $\varphi_2 = \psi_2$

$$(?),(T\exists) \quad \vdash^{IDL}_{C} (\varphi_1 \wedge \varphi_2) \leftrightarrow \varphi$$

(iv) Dual de (iii)

(v) $\quad \nu(\varphi) = \bigwedge_{n \geq 0} \chi_n$

$c(\varphi) = 0$ es un caso no posible

$c(\varphi) > 0$ Los únicos casos posibles son

- $\varphi = \langle x := \tau \rangle \psi_0$, $\varphi = [x := \tau]\psi_0$, $\varphi = \langle \alpha ; \beta \rangle \psi_0$, $\varphi = [\alpha ; \beta]\psi_0$

análogos a (i)

- $\varphi = [\alpha^*]\psi_0$

$$\nu(\varphi) = \bigwedge_{n \geq 0} \nu([\alpha^n]\psi_0) = \bigwedge_{n \geq 0} \chi_n \Longrightarrow \chi_n = \nu([\alpha^n]\psi_0) \; n \geq 0$$

Tomando $\psi = \psi_0$ tenemos $(T\exists) \quad \vdash^{IDL}_{C} \quad [\alpha^*]\psi \leftrightarrow \varphi$

(vi) Dual de (v)

(vii) $\quad \nu(\varphi) = \exists x \chi$

$c(\varphi) = 0$ es un caso no posible

$c(\varphi) > 0$ Los únicos casos posibles son

· $\varphi = \exists x \psi_o$

$$\gamma(\varphi) = \exists x\, \gamma(\psi_o) = \exists x \chi \quad \Longrightarrow \quad \gamma(\psi_o) = \chi$$

y tomando $\psi = \psi_o$ $\quad (T\exists) \quad \vdash^{IDL}_{C} \quad (\exists x \psi \leftrightarrow \varphi)$

· $\varphi = \langle x := \tau \rangle \psi_o$, $\varphi = [x := \tau] \psi_o$, $\varphi = \langle \alpha ; \beta \rangle \psi_o$, $\varphi = [\alpha ; \beta] \psi_o$

análogos a (i)

(viii) Dual de (vii)

Notese que todas las fórmulas que se concluye que existen son sentencias si φ lo es.

Teorema 1.4.2.- Completitud de $\vdash^{IDL}$

Sea $\Phi \subseteq DFS_d$ a lo sumo numerable y $\varphi \in DFS_d$. Se verifica

$$\Phi \models \varphi \quad \Longrightarrow \quad \Phi \vdash^{IDL} \varphi$$

Demostración:

Partiremos de suponer que $\Phi \not\vdash^{IDL} \varphi$ y construiremos una estructura $\mathfrak{D}$ tal que $\mathfrak{D} \models \Phi$ y $\mathfrak{D} \not\models \varphi$

Para ello aplicaremos el teorema de existencia de modelos para $L_{\omega_1 \omega}$

Definición 1.4.5.- Propiedad de consistencia

Una propiedad de consistencia S es una familia de conjuntos de sentencias de $L_{\omega_1 \omega}$ (C)(fórmulas de $L_{\omega_1 \omega}$ (C) sin variables libres) tal que para todo $s \in S$ se verifican las siguientes propiedades:

(c1) - Coherencia: Para cualquier formula atómica $\chi \in L_{\omega_1 \omega}$ (C)

$$o \quad \chi \notin s \quad o \quad \neg\chi \notin s$$

(c2) $\neg$- Propiedad: Si $\neg\chi \in s$ entonces $s \cup \{\chi_\neg\} \in S$

(c3) $\wedge$- Propiedad: Si $\bigwedge_{n<m} \chi_n \in s$, $0 < m \leq \omega$ entonces

$s \cup \{\chi_n\} \in S$ para todo n $0 \leq n < m$

(c4) $\forall$- Propiedad: Si $\forall x \chi \in s$ entonces $s \cup \{\chi[c/x]\} \in S$

para cualquier $c \in C$

(c5) $\vee$- Propiedad: Si $\bigvee_{n<m} \chi_n \in s$, $0 < m \leq \omega$ entonces

para algún $n < m$ $s \cup \{\chi_n\} \in S$

(c6) $\exists$- Propiedad: Si $\exists x \chi \in s$ entonces $s \cup \{\chi[c/x]\} \in S$

para algún $c \in C$

(c7) - Propiedades de igualdad:

Para cualquier término sin variables σ y $c,d \in C$:

(a) Si $(c \doteq d) \in s$ entonces $s \cup \{(d \doteq c)\} \in S$

(b) Si $(c \doteq \sigma)$, $\chi[\sigma/x] \in s$ para alguna formula atómica o atómica negada entonces $s \cup \{\chi[c/x]\} \in S$

(c) Para algún $e \in C$ $s \cup \{(e \doteq \sigma)\} \in S$

El teorema de existencia de modelos se enuncia:

Teorema de existencia de modelos.-

Si S es una propiedad de consistencia y $s \in S$ entonces s tiene un modelo.

La demostración puede encontrarse en (26).

Para aplicarlo hemos de definir una propiedad de consistencia adecuada.

Definimos S como la familia de todos los conjuntos s de sentencias de $L_{\omega_1\omega}(C)$ de la forma

$s = \nu(\Psi) = \{\nu(\psi)/\ \psi \in \Psi\}$ siendo Ψ un conjunto de sentencias de DFS(C) que verifique:

(a) Hay infinitas constantes de C que no aparecen en Ψ

(b) $\mathrm{Con}_C^{IDL}\ \Psi$

Lema 1.4.6.- Lema de consistencia

S es una propiedad de consistencia.

Demostración:

Sea $s \in S$, $s = \nu(\Psi)$

(c1) Sea χ atómica

Si $\chi \in s$ y $\neg\chi \in s$ entonces existen $\varphi_1, \varphi_2 \in \Psi$ con

$\nu(\varphi_1) = \chi$ $\qquad \vdash_C^{IDL} \varphi_1 \leftrightarrow \chi$

Por el lema 1.4.5 (i)

$\nu(\varphi_2) = \neg\chi$ $\qquad \vdash_C^{IDL} \varphi_2 \leftrightarrow \neg\chi$

Luego (MP), (T∃) $\quad \Psi \vdash_C^{IDL} \chi \wedge \neg\chi$ y no $\mathrm{Con}_C^{IDL}\ \Psi$

(c2) $\neg\chi \in s$, luego existe $\varphi \in \Psi$ con $\nu(\varphi) = \neg\chi$

Por el lema 1.4.5 (ii) existe ψ sentencia de DFS(C) con $\nu(\psi) = \chi\neg$

y $\vdash_C^{IDL} \psi \leftrightarrow \varphi$ Como además $\mathrm{Con}_C^{IDL}\ \Psi$ tenemos

$\mathrm{Con}_C^{IDL}(\Psi \cup \{\psi\}) \Longrightarrow s \cup \{\chi\neg\} \in S$

(c3) $\bigwedge_{n<m} \chi_n \in s$

Si m finito podemos suponer, sin perdida de generalidad, que m = 2

$\chi_1 \wedge \chi_2 \in s \Longrightarrow$ existe $\varphi \in \Psi$ con $\nu(\varphi) = \chi_1 \wedge \chi_2$

Usando el el lema 1.4.5 (iii) obtenemos

$$s \cup \{\chi_1\} \in S$$

$$s \cup \{\chi_2\} \in S$$

Si m = ω Existe $\varphi \in \Psi$ con $\nu(\varphi) = \bigwedge_{n\geq 0} \chi_n$

Por el lema 1.4.5 (v) existe $[\alpha^*]\psi$ sentencia de DFS(C) tal que

$\nu([\alpha^n]\psi) = \chi_n$ y $\vdash^{IDL}_{C}$ $[\alpha^*]\psi \leftrightarrow \varphi$

Como, además, con $(*')$ $\vdash^{IDL}_{C}$ $[\alpha^*]\psi \longrightarrow [\alpha^n]\psi$ para cualquier n

tenemos $Con^{IDL}_{C}(\Psi \cup \{[\alpha^n]\psi\})$, luego $s \cup \{\chi_n\} \in S$ para cualquier $n \in N$

(c4) $\forall x\, \chi \in s$, luego existe $\varphi \in \Psi$ con $\nu(\varphi) = \forall x \chi$

Por el lema 1.4.5(viii) existe ψ sentencia de DFS(C) con $\nu(\psi) = \chi$

y $\vdash^{IDL}_{C} \forall x \psi \leftrightarrow \varphi$

Tenemos $\Psi \vdash^{IDL}_{C} \varphi$

(T∃),(MP) $\Psi \vdash^{IDL}_{C} \forall x \psi$

(MP),(T∃) $\Psi \vdash^{IDL}_{C} \psi[c/x]$ para cualquier $c \in C$

Obtenemos $Con^{IDL}_{C}(\Psi \cup \{\psi[c/x]\})$

Por el lema 1.4.4 $\nu(\psi[c/x]) = \chi[c/x]$ y concluimos $s \cup \{\chi[c/x]\} \in S$

para cualquier $c \in C$

(c5) $\bigvee_{n<m} \chi_n \in s$

Si m finito es dual a (c3)

Si $m = \omega$ existe $\varphi \in \Psi$ con $\nu(\varphi) = \bigvee_{n\geq 0} \chi_n$

Por el lema 1.4.5 (vi) existe $\langle\alpha^*\rangle\psi$ sentencia de DFS(C) tal que

$\nu(\langle\alpha^n\rangle\psi) = \chi_n$ para cualquier n y $\vdash^{IDL}_{C}$ $\langle\alpha^*\rangle\psi \leftrightarrow \varphi$

Si para todo n $s \cup \{\chi_n\} \notin S$ tenemos

$\Psi \cup \{<\alpha^n>\Psi\} \vdash_C^{IDL}$ **false** para todo n, luego para todo n

$$\Psi \vdash_C^{IDL} <\alpha^n>\Psi \longrightarrow \underline{\text{false}}$$

$$(\infty') \Psi \vdash_C^{IDL} <\alpha^*>\Psi \longrightarrow \underline{\text{false}}$$

$$(MP), (T\exists) \Psi \vdash_C^{IDL} \varphi \longrightarrow \underline{\text{false}}$$

Luego no $Con_C^{IDL}\ \Psi$ y llegamos a contradicción.

(c6) $\exists x \chi \in s$, luego existe $\varphi \in \Psi$ con $\nu(\varphi) = \exists x \chi$

Por el lema 1.5.5 (vii) existe $\exists x \Psi$ sentencia de DFS(C) con

$$\nu(\Psi) = \chi \quad y \quad \vdash_C^{IDL} \exists x \Psi \leftrightarrow \varphi$$

Por el lema 1.4.4 $\nu(\Psi[c/x]) = \chi[c/x]$

Si $s \cup \{\Psi[c/x]\} \notin S$ para cualquier $c \in C$ tenemos

$\Psi \cup \{\Psi[c/x]\} \vdash_C^{IDL}$ **false** , luego

$$\Psi \vdash_C^{IDL} \Psi[c/x] \longrightarrow \underline{\text{false}}$$

Por como hemos escogido Ψ podemos elegir $c \in C$ con c no apareciendo ni en Ψ ni en $\exists x \Psi$ y para dicha c puedo aplicar la regla (I∃)

$$(I\exists)\ \Psi \vdash_C^{IDL} \exists x \Psi \longrightarrow \underline{\text{false}}$$

$$(MP)\ \Psi \vdash_C^{IDL} \varphi \longrightarrow \underline{\text{false}}$$ llegando a contradicción con $Con_C^{IDL}\ \Psi$

(c7) Para cualquier término sin variables σ y $c, d \in C$

(a) $(c \doteq d) \in s$, luego existe $\varphi \in \Psi$ con $\nu(\varphi) = (c \doteq d)$

Por el lema 1.4.5 (i) $\vdash_C^{IDL} \varphi \leftrightarrow (c \doteq d)$

$(MP), (T\exists)$ $\vdash_C^{IDL} \varphi \leftrightarrow (d \doteq c)$

Luego $\Psi \vdash_C^{IDL} (d \doteq c)$ y por tanto $Con_C^{IDL}(\Psi \cup \{(d \doteq c)\})$

y como $\nu(d \doteq c) = (d \doteq c)$, $s \cup \{(d \doteq c)\} \in S$

(b) $(c \doteq d)$, $\chi[\sigma/x] \in s$ existen $\varphi_1, \varphi_2 \in \Psi$ con

$\gamma(\varphi_1) = (c \doteq \sigma) \Longrightarrow$ (lema 1.4.5 (i)) $\vdash^{IDL}_{C} (c \doteq \sigma) \longleftrightarrow \varphi_1$

$\gamma(\varphi_2) = \chi[\sigma/x] \Longrightarrow$ (lema 1.4.5 (i)) $\vdash^{IDL}_{C} \chi[\sigma/x] \longleftrightarrow \varphi_2$

(T∃),(MP) $\vdash^{IDL}_{C} (\varphi_1 \wedge \varphi_2) \longrightarrow ((c \doteq \sigma) \wedge \chi[\sigma/x])$

(T∃) $\vdash^{IDL}_{C} ((c \doteq \sigma) \wedge \chi[\sigma/x]) \longrightarrow \chi[c/x]$

y con (MP) obtenemos $\vdash^{IDL}_{C} \varphi_1 \wedge \varphi_2 \rightarrow \chi[c/x]$. Luego $Con^{IDL}_{C}(\Psi \cup \{\chi[c/x]\})$

y como $\gamma(\chi[c/x]) = \chi[c/x]$ $\quad s \cup \{\chi[c/x]\} \in S$

(c) Supongamos que para cualquier $e \in C$ $\quad s \cup \{(e \doteq \sigma)\} \notin S \Longrightarrow$

$\Psi \cup \{(e \doteq \sigma)\} \vdash^{IDL}_{C}$ **false** $\Longrightarrow$

$\Psi \vdash^{IDL}_{C} (x \doteq \sigma)[e/x] \longrightarrow$ **false**

Escojemos $e \in C$ con e no apareciendo ni en Ψ ni en σ y aplicamos (I∃)

$\Psi \vdash^{IDL}_{C} \exists x(x \doteq \sigma) \longrightarrow$ **false**

Como por (T∃) $\vdash^{IDL}_{C} \exists x(x \doteq \sigma)$ concluiimos no $Con^{IDL}_{C}\ \Psi$

llegando a contradicción

El teorema se demuestra ahora con los siguientes pasos:

(1) $\Phi \vdash^{IDL} \varphi \Longrightarrow \Phi^{\forall} \vdash^{IDL} \varphi^{\forall}$ (donde $\varphi^{\forall}$ representa el cierre universal: cuantificar universalmente todas las variables libres. $\Phi^{\forall}$ es la generalización obvia a conjuntos de fórmulas)

(1) es cierto ya que

$\Phi^{\forall} \vdash^{IDL} \varphi^{\forall} \Longrightarrow \Phi \vdash^{IDL} \varphi$ gracias a que para cualquier $\psi \in DFS_d$:

$\psi \vdash^{IDL} \psi^{\forall}$ aplicando reiteradamente (G2')

$\psi^{\forall} \vdash^{IDL} \psi$ aplicando reiteradadamente (MP) y axiomas de (T$\exists$) de la forma $\forall x \psi \longrightarrow \psi$

(2) $\Phi^{\forall} \not\vdash^{IDL} \varphi^{\forall} \Longrightarrow \Psi = \Phi^{\forall} \cup \{ \neg \varphi^{\forall} \}$ cumple $Con^{IDL} \Psi$

(3) $Con^{IDL} \Psi \Longrightarrow Con_C^{IDL} \Psi$

ya que Ψ es un conjunto de sentencias, luego hay infinitas variables w que no aparecen libres en Ψ . Si ocurriese $\Psi \vdash_C^{IDL} \underline{false}$ podriamos reemplazar las constantes de C usadas en la demostración por variables, obteniendo $\Psi \vdash^{IDL} \underline{false}$, que contradice $Con^{IDL} \Psi$

(4) $s = \gamma(\Psi) \in S$ por (2) y (3). Por ser S una propiedad de consistencia concluimos Sat s . Como la traducción γ respeta la semántica (lema 1.4.2), tambien se cumple Sat Ψ , esto es, obtenemos una estructura $\mathfrak{D} \in STR_d$ tal que $\mathfrak{D} \models \Phi^{\forall} \cup \{ \neg \varphi^{\forall} \}$

En esta $\mathfrak{D}$ ocurre $\mathfrak{D} \models \Phi$, $\mathfrak{D} \not\models \varphi$, luego $\Phi \not\models \varphi$.

1.5.- Estructuras aritméticas.

En las secciones anteriores hemos presentado un cálculo completo IDL para la lógica DL. IDL utiliza una regla con infinitas premisas (ω), lo que lo convierte en poco útil para su utilización práctica.

Es posible construir un cálculo finitario completo para ciertas estructuras especiales que llamaremos estructuras aritméticas (porque contienen a los números naturales).

Los números naturales y sus operaciones serán usados en las fórmulas de primer orden para "contar" el número de veces que es ejecutado α en α^*. Para cada $\varphi \in DFS_d$ podremos construir una fórmula de primer orden $\varphi' \in FS_d$ equivalente a φ sobre $\mathfrak{D}$. Esta propiedad, llamada de expresividad, es decisiva en la demostración del teorema de completitud para este cálculo.

Necesitaremos tomar como axiomas adicionales al cálculo todas las fórmulas de primer orden válidas en la estructura.

Obtendremos el teorema de completitud aritmética de Harel, que generaliza el teorema de Cook para la lógica de Hoare sobre una estructura expresiva.

Definición 1.5.1.- Estructuras aritméticas

$\mathfrak{D} \in STR_d$ es una estructura aritmética si y solo si:

· Su dominio incluye una copia isomorfa de los números naturales: $N \subseteq D$

· En d hay símbolos 0 (constante), suc, comp (de funcion), Nat, $<$ (predicados) cuyas interpretaciones cumplen:

$Nat^{\mathfrak{D}} = N$

$0^{\mathfrak{D}} = 0^N$

$suc^{\mathfrak{D}} \restriction N = suc^N$

$<^{\mathfrak{D}} = <^N$

$comp^{\mathfrak{D}} \restriction D \times N \longrightarrow D$ es tal que dados $n \in N$ y $a_0, \ldots, a_n \in D$ existe $a \in D$ tal que $comp^{\mathfrak{D}}(a,i) = a_i$ para $0 \leq i \leq n$

Un ejemplo de estructura aritmética es:

Sea $\mathfrak{N} = (N, 0^N, suc^N, +^N, \cdot^N, \leq^N)$ modelo standard de la aritmética.

Falta definir la función de decodificación comp.

Sea $\Pi^2 : N \times N \xrightarrow{biy} N$ definida por $\Pi^2(a,b) = \frac{1}{2}(a+b)(a+b+1) + a$

Sean Π^2_1, Π^2_2 las proyecciones de la inversa de Π^2 y sea

$\delta : N \times N \longrightarrow N$ definida por $\delta(a,i) = \Pi^2_1(a) \bmod 1 + (i+1).\Pi^2_2(a)$

Usando el teorema chino de los restos puede demostrarse que

$\forall n, a_0, \ldots, a_n \in N, \exists a \in N \ \forall i \leq n \ \ \delta(a,i) = a_i$ (función β de Gödel).

δ es definible en $\mathfrak{N}$ y permite ampliar $\mathfrak{N}$ a una estructura aritmética.

Teorema 1.5.1.-

Sea $\mathfrak{D} \in STR_d$. Si $\mathfrak{D}$ es infinita puede ampliarse a una estructura aritmética.

Demostración:

Sin perdida de generalidad podemos suponer $N \subseteq D$

Sea $K : D^* \longrightarrow D$ una biyección ($D^* = \bigcup_{n<\omega} D^n$). Añadimos a $\mathfrak{D}$ la función $comp^{\mathfrak{D}}$ poniendo:

$$comp^{\mathfrak{D}}(a,i) = \begin{cases} a_i \text{ si } K^{-1}(a) = a_0 \cdot \ldots \cdot a_n, & n \geq i \\ 1 \text{ en otro caso con } 1 \in D \text{ fijo} \end{cases}$$

$Nat^{\mathfrak{D}}$, $<^{\mathfrak{D}}$, $0^{\mathfrak{D}}$, $suc^{\mathfrak{D}}$ son fáciles de añadir.

Definición 1.5.2.-

$\mathfrak{D}$ es expresiva con respecto a PS_d si y solo si para cada $\alpha \in PS_d$ con variables $\bar{x}$ existe una fórmula $\int_\alpha \in FS_d$ con variables libres $\bar{u}$, $\bar{v}$ que define $\alpha^{\mathfrak{D}}$ en $\mathfrak{D}$:

Para $s,t \in W$ $\quad s \alpha^{\mathfrak{D}} t \iff \mathfrak{D} \models \int_\alpha [s(\bar{x})/\bar{u},\ t(\bar{x})/\bar{v}]$

Teorema 1.5.2.- Expresividad

Si $\mathfrak{D}$ es aritmética entonces $\mathfrak{D}$ es expresiva con respecto a PS_d y cada fórmula de DFS_d es equivalente en $\mathfrak{D}$ a una fórmula de FS_d.

Demostración:

Como abreviaturas usaremos en la demostración;

$$(x)_i = comp(x,i)$$

$$i = suc^i(0)$$

$$(w)_i = \bar{u} \text{ es } (u_0 = ((w)_i)_0 \wedge \ldots\ldots \wedge u_n = ((w)_i)_n$$

La demostración para probar que $\mathfrak{D}$ es expresiva con respecto a PS_d se realiza por inducción sobre la estructura de α

· $\alpha = (x_i := \tau)$ $\quad \int_\alpha = (v_1 \doteq u_1 \wedge \ldots \wedge v_{i-1} \doteq u_{i-1} \wedge v_{i+1} \doteq u_{i+1} \wedge \ldots \wedge v_n \doteq u_n$

$\wedge v_i \doteq \tau[\bar{u}/\bar{x}]$)

· $\alpha = \varphi$? Por hipótesis de inducción existe $\varphi' \in FS_d$ con $\mathfrak{D} \models (\varphi \leftrightarrow \varphi')$

Tomamos $\quad \int_\alpha = \varphi'[\bar{u}/\bar{x}] \wedge \bar{v} \doteq \bar{u}$

· $\alpha = \beta \cup \beta'$ $\quad \int_\alpha = \int_\beta \vee \int_{\beta'}$

· $\alpha = \beta ; \beta'$ $\quad \int_\alpha = \exists \bar{w} \, (\int_\beta (\bar{u}, \bar{w}) \wedge \int_{\beta'} (\bar{w}, \bar{v}))$

Aquí $\int_\beta (\bar{u}, \bar{w})$, $\int_{\beta'} (\bar{w}, \bar{v})$ indican la sustitución natural.

· $\alpha = \beta^*$ Llamamos $\underline{\text{cómputo}}_\beta (w, \bar{u}, \bar{v}, x) = (Nat(x) \wedge (w)_0 \doteq \bar{u} \wedge (w)_x \doteq \bar{v} \wedge$

$\forall y ((Nat(y) \wedge y < x) \longrightarrow \int_\beta ((w)_y, (w)_{suc(y)})))$

$\int_\alpha = \exists z \exists w \; \underline{\text{cómputo}}_\beta (w, \bar{u}, \bar{v}, z)$

$s(\beta^*)^{\mathfrak{D}} t \iff$ existe m y existen $s = r_0, \ldots, r_m = t$ con $r_0 \beta^{\mathfrak{D}} r_1 \beta^{\mathfrak{D}} \ldots \beta^{\mathfrak{D}} r_m$

$\iff$ (Por hipótesis de inducción) existe m y existen $s = r_0, \ldots, r_m = t$ con

$\mathfrak{D} \models \bigwedge_{0 \leq j < m} \int_\beta [r_j(\bar{x})/\bar{u}, r_{suc(j)}(\bar{x})/\bar{v}] \iff$ Con la función comp

$\mathfrak{D} \models \int_\alpha [s(\bar{x})/\bar{u}, t(\bar{x})/\bar{v}]$

La segunda parte del teorema afirma que para cualquier $\varphi \in DFS_d$ existe una fórmula $\varphi' \in FS_d$ tal que $\mathfrak{D} \models (\varphi \leftrightarrow \varphi')$

Procedemos por inducción sobre la estructura de φ

· φ es de primer orden $\quad \varphi' = \varphi$

· $\varphi = \neg \psi$ $\quad \varphi' = \neg \psi'$ y $\mathfrak{D} \models (\varphi \leftrightarrow \varphi')$

· $\varphi = \psi \wedge \chi$ $\quad \varphi' = \psi' \wedge \chi'$ y $\quad \mathfrak{D} \models (\varphi \leftrightarrow \varphi')$

· $\varphi = \exists x \psi$ $\quad \varphi' = \exists x \psi'$ y $\mathfrak{D} \models (\varphi \leftrightarrow \varphi')$

· $\varphi = \langle \alpha \rangle \psi$ Sea $\bar{x} = var(\alpha)$ $\quad \varphi' = \exists \bar{v} \, (\int_\alpha [\bar{x}/\bar{u}] \wedge \psi'[\bar{v}/\bar{x}])$

1.6.- El cálculo P. Teorema de completitud de Harel.

Definición 1.6.1.- Cálculo P para DL

Axiomas:

(T∃) Todas las instancias de tautologias del cálculo-proposicional y axiomas para el ∃ y la igualdad resultantes de sustituir consistentemente variables proposicionales o de fórmulas por fórmulas de DFS_d.

(:=) $\langle x := \tau \rangle \varphi \longleftrightarrow \varphi[\tau/x]$

(?) $\langle \psi ? \rangle \varphi \longleftrightarrow \psi \wedge \varphi$

(;) $\langle \alpha ; \beta \rangle \varphi \longleftrightarrow \langle \alpha \rangle \langle \beta \rangle \varphi$

(∪) $\langle \alpha \cup \beta \rangle \varphi \longleftrightarrow \langle \alpha \rangle \varphi \vee \langle \beta \rangle \varphi$

Reglas:

(MP) $$\frac{\varphi, \quad \varphi \longrightarrow \psi}{\psi}$$

(G1) $$\frac{\varphi \longrightarrow \psi}{\langle \alpha \rangle \varphi \longrightarrow \langle \alpha \rangle \psi}$$

(G2) $$\frac{\varphi \longrightarrow \psi}{\exists x \varphi \longrightarrow \exists x \psi}$$

(I) $$\frac{\varphi \longrightarrow [\alpha]\varphi}{\varphi \longrightarrow [\alpha^*]\varphi}$$

(C) $$\frac{(Nat(z) \wedge \varphi[suc(z)/z]) \longrightarrow \langle \alpha \rangle \varphi}{(Nat(z) \wedge \varphi) \longrightarrow \langle \alpha^* \rangle \varphi[0/z]}$$

$\varphi \in FS_d$, z libre en φ

$z \notin var(\alpha)$

La regla (I) se llama regla de invariancia y (C) regla de convergencia.

Th ($\mathfrak{D}$) es la teoría de primer orden de $\mathfrak{D}$

$Th(\mathfrak{D}) = \{ \varphi \in FS_d / \ \mathfrak{D} \models \varphi \}$

Teorema 1.6.1.- Corrección

Para cualquier estructura aritmética $\mathfrak{D}$ y fórmula $\varphi \in DFS_d$

$Th(\mathfrak{D}) \overset{P}{\vdash} \varphi \implies \mathfrak{D} \models \varphi$

Demostración:

Basta probar que los axiomas son válidos en $\mathfrak{D}$ y que las reglas conservan la validez en $\mathfrak{D}$.

Por el lema 1.3.2 los axiomas son válidos. En el teorema 1.4.1 tenemos una prueba de la corrección de (G1).

Queda probar la corrección de (I) y (C)

Corrección de (I)

Para cualquier $\varphi \in DFS_d$ y $\alpha \in PS_d$

$\mathfrak{D} \models (\varphi \to [\alpha] \varphi)$ entonces $\mathfrak{D} \models (\varphi \to [\alpha^*] \varphi)$

Sea $s \in W_{\mathfrak{D}}$ y supongamos $\mathfrak{D} \models_s \varphi$

$\mathfrak{D} \models_s [\alpha^*] \varphi \iff$ para todo n $\mathfrak{D} \models_s [\alpha^n] \varphi$

y esto vamos a probarlo por inducción sobre n

n=0 $[\alpha^0] \varphi = \varphi$ y por hipótesis $\mathfrak{D} \models_s \varphi$

Supuesto para n veámoslo para n+1

Por hipótesis $\mathfrak{D} \models \varphi \to [\alpha] \varphi$

Por la corrección de (G1) $\mathfrak{D} \models [\alpha^n] \varphi \to [\alpha^n][\alpha] \varphi$

Por la corrección de (;) $\mathfrak{D} \models [\alpha^n] \varphi \to [\alpha^{n+1}] \varphi$

y como por hipótesis de inducción $\mathfrak{D} \models_s [\alpha^n]\varphi$ concluimos $\mathfrak{D} \models_s [\alpha^{n+1}]\varphi$

Corrección de (C)

Para cualquier $\varphi(z) \in FS_d$, $\alpha \in PS_d$ y $z \notin var(\alpha)$

$\mathfrak{D} \models (Nat(z) \wedge \varphi[suc(z)/z] \longrightarrow \langle\alpha\rangle\varphi)$ entonces

$$\mathfrak{D} \models (Nat(z) \wedge \varphi) \longrightarrow \langle\alpha^*\rangle\varphi[0/z]$$

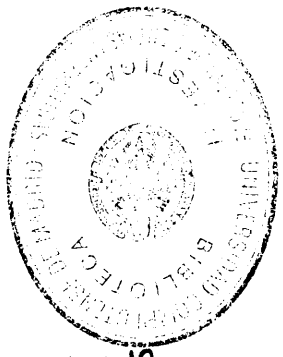
Sea $s \in W_{\mathfrak{D}}$ y supongamos $\mathfrak{D} \models_s (Nat(z) \wedge \varphi)$

Lo probaremos por inducción sobre el valor de z en s m=s(z)

m=0

$\mathfrak{D} \models_s \varphi \Longrightarrow (s(z)=0)\ \mathfrak{D} \models_s \varphi[0/z] \Longrightarrow \mathfrak{D} \models_s \langle\alpha^0\rangle\varphi[0/z] \Longrightarrow$

$\mathfrak{D} \models_s \langle\alpha^*\rangle\varphi[0/z]$



Supuesto para m = k veámoslo para m = suc(k)

$\mathfrak{D} \models_s (Nat(z) \wedge \varphi)$ por hipótesis. Tomando $s'=s[k/z]$

tenemos $\mathfrak{D} \models_{s'} (Nat(z) \wedge \varphi[suc(z)/z])$

Como $\mathfrak{D} \models (Nat(z) \wedge \varphi[suc(z)/z] \longrightarrow \langle\alpha\rangle\varphi)$ obtenemos $\mathfrak{D} \models_{s'} \langle\alpha\rangle\varphi$

luego existe $t \in W_{\mathfrak{D}}$ con $s'\alpha^{\mathfrak{D}}t$ y $\mathfrak{D} \models_t \varphi$

Como $z \notin var(\alpha)$ $t(z) = s'(z) = k$ podemos aplicar la hipótesis de inducción al estado t.

Asi existe $t \in W_{\mathfrak{D}}$ con $s'\alpha^{\mathfrak{D}}t$ y $\mathfrak{D} \models_t \langle\alpha^*\rangle\varphi[0/z] \Longrightarrow$

$\mathfrak{D} \models_{s'} \langle\alpha\rangle\langle\alpha^*\rangle\varphi[0/z] \Longrightarrow \mathfrak{D} \models_{s'} \langle\alpha^*\rangle\varphi[0/z] \Longrightarrow$

(z no aparece en $\varphi[0/z]$) $\mathfrak{D} \models_s \langle\alpha^*\rangle\varphi[0/z]$

Con lo que concluimos que el cálculo P es correcto con respecto a cualquier estructura aritmética. Más aun, lo único que hasta ahora hemos usado de la hipótesis de aritmeticidad es que $0^{\mathfrak{D}}$ y $suc^{\mathfrak{D}}$ gene-

ran una copia isomorfa de N dentro de $\mathfrak{D}$ (para razonar por inducción en la demostración de corrección de (C)).

Lema 1.6.1.-

Las siguientes son reglas derivadas de P

(G1') $\dfrac{\varphi \longrightarrow \psi}{[\alpha]\varphi \longrightarrow [\alpha]\psi}$ (G2') $\dfrac{\varphi \longrightarrow \psi}{\forall x\varphi \longrightarrow \forall x\psi}$

(I') $\dfrac{\psi \longrightarrow \varphi \ , \ \varphi \longrightarrow [\alpha]\varphi \ , \ \varphi \longrightarrow \chi}{\psi \longrightarrow [\alpha^*]\chi}$

(C') $\dfrac{\psi \longrightarrow \exists z(\mathrm{Nat}(z) \wedge \varphi\ ),\ (\mathrm{Nat}(z) \wedge \varphi[\mathrm{suc}(z)/z]\ ) \longrightarrow \langle\alpha\rangle\varphi \ , \ \varphi[0/z] \longrightarrow \chi}{\psi \longrightarrow \langle\alpha^*\rangle\chi}$

$z \notin \mathrm{var}(\alpha)$, φ de primer orden y z libre en φ

Demostración:

(G1')

$\vdash^{P} \varphi \longrightarrow \psi$

(MP),(T$\exists$) $\vdash^{P} \neg\psi \longrightarrow \neg\varphi$

(G1) $\vdash^{P} \langle\alpha\rangle\neg\psi \longrightarrow \langle\alpha\rangle\neg\varphi$

(MP),(T$\exists$) $\vdash^{P} [\alpha]\varphi \longrightarrow [\alpha]\psi$

(G2') se demuestra análogamente.

(I')

$\vdash^{P} \varphi \longrightarrow [\alpha]\varphi$

(I) $\vdash^{P} \varphi \longrightarrow [\alpha^*]\varphi$ (1)

$\vdash^{P} \psi \longrightarrow \varphi$

(1)(T$\exists$)(MP) $\vdash^{P} \psi \longrightarrow [\alpha^*]\varphi$ (2)

$\vdash^{P} \varphi \longrightarrow \chi$

(G1') $\vdash^{P} [\alpha^*]\varphi \longrightarrow [\alpha^*]\chi$

(2)(T∃)(MP) $\vdash^{P} \psi \longrightarrow [\alpha^*]\chi$

(C') se demuestra análogamente

Lema 1.6.2.- Lema del invariante

Sea $\mathfrak{D}$ una estructura aritmética.

Para cualquier $\alpha \in PS_d$ y $\psi, \chi \in DFS_d$

si $\mathfrak{D} \models (\psi \longrightarrow [\alpha^*]\chi)$ entonces existe una fórmula φ de primer orden tal que $\mathfrak{D} \models (\psi \longrightarrow \varphi)$, $\mathfrak{D} \models (\varphi \longrightarrow [\alpha]\varphi)$, $\mathfrak{D} \models (\varphi \longrightarrow \chi)$

Demostración:

Por el teorema 1.5.2 existe φ de primer orden con $\mathfrak{D} \models \varphi \longleftrightarrow [\alpha^*]\chi$

Como $\mathfrak{D} \models (\psi \longrightarrow [\alpha^*]\chi)$ por hipótesis tenemos $\mathfrak{D} \models \psi \longrightarrow \varphi$

Como $\models [\alpha^*]\chi \longrightarrow [\alpha][\alpha^*]\chi$ tenemos que $\mathfrak{D} \models \varphi \longrightarrow [\alpha]\varphi$

Como $\mathfrak{D} \models \varphi \longrightarrow [\alpha^*]\chi$ en particular $\mathfrak{D} \models \varphi \longrightarrow [\alpha^0]\chi$, luego

$\mathfrak{D} \models \varphi \longrightarrow \chi$

La fórmula φ del lema se llama invariante de α.

Teorema 1.6.2.- [ ]-Completitud

Sea $\mathfrak{D}$ una estructura aritmética.

Para cualquier $\alpha \in PS_d$ y ψ, χ de primer orden

$\mathfrak{D} \models (\psi \longrightarrow [\alpha]\chi)$ entonces $Th(\mathfrak{D}) \vdash^{P} (\psi \longrightarrow [\alpha]\chi)$

Demostración:

Por inducción sobre la estructura de α

Si α es una asignación o pregunta ? el problema se reduce por (:=') y (?') (vease lema 1.4.1) a probar una fórmula de primer orden válida

en $\mathfrak{D}$, que está disponible como hipótesis en Th($\mathfrak{D}$).

Si $\alpha = \beta \cup \beta'$

$\mathfrak{D} \models \psi \longrightarrow [\alpha]\chi \iff \mathfrak{D} \models \psi \longrightarrow [\beta]\chi$ y $\mathfrak{D} \models \psi \longrightarrow [\beta']\chi$

y ambas son demostrables por hipótesis de inducción. Con el axioma dual ($\cup'$) y (T$\exists$) queda demostrada la fórmula.

Si $\alpha = \beta ; \beta'$

$\mathfrak{D} \models \psi \longrightarrow [\alpha]\chi \iff \mathfrak{D} \models \psi \longrightarrow [\beta][\beta']\chi$

Por el teorema 1.5.2 existe φ de primer orden con $\mathfrak{D} \models \varphi \longleftrightarrow [\beta']\chi$

Ahora como $\mathfrak{D} \models \psi \longrightarrow [\beta]\varphi$ tenemos Th($\mathfrak{D}$) $\overset{P}{\vdash} \psi \longrightarrow [\beta]\varphi$ por hipótesis de inducción. Tambien por hipótesis de inducción Th($\mathfrak{D}$) $\overset{P}{\vdash} \varphi \longrightarrow [\beta']\chi$ y por (Gl')(MP)(T$\exists$) y (;') obtenemos Th($\mathfrak{D}$) $\overset{P}{\vdash} \psi \longrightarrow [\alpha]\chi$

Si $\alpha = \beta^*$ usamos el lema 1.6.2 y existe φ de primer orden tal que $\mathfrak{D} \models (\psi \longrightarrow \varphi)$, $\mathfrak{D} \models (\varphi \longrightarrow [\beta]\varphi)$, $\mathfrak{D} \models (\varphi \longrightarrow \chi)$. Por hipótesis de inducción Th($\mathfrak{D}$) $\overset{P}{\vdash} \psi \longrightarrow \varphi$, Th($\mathfrak{D}$) $\overset{P}{\vdash} \varphi \longrightarrow [\beta]\varphi$, Th($\mathfrak{D}$) $\overset{P}{\vdash} \varphi \longrightarrow \chi$. Usando (I') obtenemos Th($\mathfrak{D}$) $\overset{P}{\vdash} \psi \longrightarrow [\alpha]\chi$

Para esta demostración hemos usado los axiomas duales a los del cálculo para [ ] . Estos axiomas están demostrados en el lema 1.4.1 para el cálculo IDL, pero la demostración es válida para el cálculo P.

Lema 1.6.3.- Lema de convergencia

Sea $\mathfrak{D}$ una estructura aritmética.

Para cualquier $\alpha \in PS_d, \psi, \chi \in DFS_d$

Si $\mathfrak{D} \models (\psi \longrightarrow \langle \alpha^* \rangle \chi)$ existe una fórmula φ de primer orden con variable libre z, $z \notin var(\alpha)$ tal que

$\mathfrak{D} \models \psi \longrightarrow \exists z(\mathrm{Nat}(z) \wedge \varphi)$, $\mathfrak{D} \models (\mathrm{Nat}(z) \wedge \varphi[\mathrm{suc}(z)/z]) \longrightarrow \langle\alpha\rangle\varphi$,

$\mathfrak{D} \models \varphi[0/z] \longrightarrow \chi$

Demostración:

Basándonos en la demostración del teorema 1.5.2 podemos definir

$\varphi(z) = \exists w \exists \bar{y}$ (cómputo$_\alpha$ $(w, \bar{x}, \bar{y}, z) \wedge \chi[\bar{y}/\bar{x}]$) con $\bar{x} = \mathrm{var}(\alpha)$ y

tenemos $\mathfrak{D} \models \langle\alpha^n\rangle\chi \leftrightarrow \varphi[n/z]$ cualquiera que sea n.

Si $\mathfrak{D} \models_s \psi \longrightarrow \langle\alpha^*\rangle\chi$ entonces existe n con $\mathfrak{D} \models \psi \longrightarrow \langle\alpha^n\rangle\chi$

Luego $\mathfrak{D} \models_s \psi \longrightarrow \exists z(\mathrm{Nat}(z) \wedge \varphi)$

Evidentemente $\mathfrak{D} \models \varphi[0/z] \longrightarrow \langle\alpha^0\rangle\chi$, es decir $\mathfrak{D} \models \varphi[0/z] \longrightarrow \chi$

Si $\mathfrak{D} \models_s (\mathrm{Nat}(z) \wedge \varphi[\mathrm{suc}(z)/z])$ y $s(z)=n$ entonces $\mathfrak{D} \models_s \langle\alpha^{n+1}\rangle\chi$

Por corrección de (;) tenemos $\mathfrak{D} \models_s \langle\alpha\rangle\langle\alpha^n\rangle\chi$ y por tanto

$\mathfrak{D} \models_s \langle\alpha\rangle\varphi$ pues z no aparece en α

A la fórmula φ del teorema se le llama fórmula de convergencia de α

Teorema 1.6.3.- $\langle\ \rangle$-Completitud

Sea $\mathfrak{D}$ una estructura aritmética.

Para cualquier $\alpha \in PS_d$, $\psi, \chi \in FS_d$

si $\mathfrak{D} \models \psi \longrightarrow \langle\alpha\rangle\chi$ entonces $\mathrm{Th}(\mathfrak{D}) \vdash^{P} \psi \longrightarrow \langle\alpha\rangle\chi$

Demostración:

Inducción sobre la estructura de α

Con (:=), (?), ($\cup$), (;) y (Gl) se demuestran los casos de la asignación, pregunta, $\cup$ y ;

Si $\alpha = \beta^*$ usamos el lema 1.6.3 y la regla (C') y obtenemos el teorema.

Teorema 1.6.4.- Teorema de completitud de Harel (18)

Para cualquier $\mathfrak{D} \in STR_d$ aritmética y $\varphi \in DFS_d$

$$\mathfrak{D} \models \varphi \quad \Longrightarrow \quad Th(\mathfrak{D}) \overset{P}{\vdash} \varphi$$

Demostración:

Por el axioma (T∃) podemos suponer φ en forma normal conjuntiva. Procedemos ahora por inducción sobre m = suma de apariciones de los símbolos $\langle\ \rangle$, $[\ ]$, y cuantificadores no triviales (cuantificadores que afectan a fórmulas que no son de primer orden)

Si m = 0 φ es de primer orden, y φ está en $Th(\mathfrak{D})$.

Si m > 0 supongamos el teorema cierto para fórmulas con menos de m apariciones de $\langle\ \rangle$, $[\ ]$ y cuantificadores no triviales.

Si $\varphi = \psi \wedge \chi$ $\quad \mathfrak{D} \models \varphi \iff \mathfrak{D} \models \psi$ y $\mathfrak{D} \models \chi$

y si $Th(\mathfrak{D}) \overset{P}{\vdash} \psi$ y $Th(\mathfrak{D}) \overset{P}{\vdash} \chi \quad \Rightarrow \quad Th(\mathfrak{D}) \overset{P}{\vdash} \psi \wedge \chi$

luego el caso de la conjunción queda reducido a los demás casos.

Solo queda probar el caso de la disyunción. Sin perdida de generalidad podemos suponer que φ es de una de estas formas:

$$\psi \vee \langle\alpha\rangle\chi, \qquad \psi \vee [\alpha]\chi\ , \qquad \psi \vee \exists x \chi \quad o \quad \psi \vee \forall x \chi$$

Notese que 1) φ esta en forma normal conjuntiva

2) ψ puede ser una fórmula vacía.

En estos casos ψ y χ tienen menos de m apariciones de $\langle\ \rangle$, $[\ ]$ y cuantificadores no triviales.

Usaremos el símbolo λ para denotar $\langle\alpha\rangle$, $[\alpha]$, $\exists x$, $\forall x$ dependiendo del caso, de manera que los cuatros casos se generalizan en $\psi \vee \lambda\chi$

Ahora $\mathfrak{D} \models \psi \vee \lambda\chi \iff \mathfrak{D} \models \neg\psi \rightarrow \lambda\chi$

Por el teorema 1.5.2 existen ψ' y χ' de primer orden tales que

$$\mathfrak{D} \models \psi \leftrightarrow \psi'$$

$$\mathfrak{D} \models \chi \leftrightarrow \chi'$$

Luego $\mathfrak{D} \models \neg\psi' \rightarrow \lambda\chi'$

Ahora si $\lambda = \langle\alpha\rangle$ usamos el teorema 1.6.3 ($\langle\ \rangle$ -Completitud)

$\lambda = [\alpha]$ usamos el teorema 1.6.2 ($[\ ]$ -Completitud)

$\lambda = \exists x$, $\lambda = \forall x$ entonces $\neg\psi' \rightarrow \lambda\chi'$ es de primer orden y está en $Th(\mathfrak{D})$

y tenemos $Th(\mathfrak{D}) \vdash^{P} \neg\psi' \rightarrow \lambda\chi'$ (1)

Además $\mathfrak{D} \models \neg\psi \rightarrow \neg\psi'$

$\mathfrak{D} \models \chi' \rightarrow \chi$

Como en ambos casos ψ' y χ' son de primer orden se puede aplicar la hipótesis de inducción y tenemos

$Th(\mathfrak{D}) \vdash^{P} \neg\psi \rightarrow \neg\psi'$ (2)

$Th(\mathfrak{D}) \vdash^{P} \chi' \rightarrow \chi$ (3)

Con (3) y (G1), (G2) o (G1'), (G2') según el caso

$Th(\mathfrak{D}) \vdash^{P} \lambda\chi' \rightarrow \lambda\chi$ (4)

Con (1), (2), (T∃) y (MP) tenemos

$Th(\mathfrak{D}) \vdash^{P} \neg\psi \rightarrow \lambda\chi'$ (5)

Y con (4), (5), (T∃) y (MP)

$Th(\mathfrak{D}) \vdash^{P} \neg\psi \rightarrow \lambda\chi$

Con (MP) y (T∃) concluimos

$$Th(\mathcal{D}) \vdash^{P} \psi \vee \lambda\chi$$

y el teorema queda demostrado.

El cálculo P es una extensión del cálculo de Hoare H, propuesto para la corrección parcial de programas (cfr. (22)). Cook en (9) investigó la completitud del sistema de axiomas H.

Para Cook un cálculo $\mathcal{C}$, correcto, se dice completo para L' relativo a L (L,L' lenguajes con $L \subseteq L'$) si para cualquier estructura expresiva para L' con respecto a L, $\mathcal{C}$ es completo para L' con respecto a $\mathcal{D}$.

Si llamamos $L_H = \{ \varphi \longrightarrow [\alpha]\psi \ / \ \varphi, \psi$ fórmulas del lenguaje $L\}$ con L lenguaje de primer orden, obtenemos:

Teorema de Cook.-

H es completo para L_H relativo a L

El teorema es en esencia el teorema 1.6.2 de $[\,]$-Completitud.

Es de destacar que en las pruebas de los teoremas 1.6.2 ($[\,]$-Completitud), 1.6.3 ($<>$-Completitud) y 1.6.4 (Completitud) es decisivo el concepto de expresividad y el teorema 1.5.2 de expresividad. Esta técnica será repetida para probar otros teoremas de completitud en otras lógicas de programas.

CAPITULO 2: LOGICAS DE LA PROGRAMACION CON SEMANTICA NO STANDARD.

2.1.- Introducción.

Como vimos en el capítulo 1 para la lógica dinámica hay cálculos infinitarios completos y cálculos finitarios relativamente completos con respecto a estructuras aritméticas. Sin embargo no existen cálculos finitarios completos (en el sentido absoluto), ya que la colección de sentencias válidas de DL es Π_1^1-completa (cfr. (18)). Para evitar estos problemas algunos autores han propuesto trabajar con semántica no standard.

Estas nuevas semánticas afectan a la iteración * . Un ejemplo de esto puede verse en el trabajo de Hájek (17). Usando modelos no standard de la teoría de números se pueden obtener iteraciones finitas en nuestras estructuras, pero que corresponden intuitivamente a iteraciones infinitas.

Hájek demostró un teorema de completitud con semántica no standard semejante al de completitud de Harel. Las estructuras con las que trabaja son modelos de la aritmética de Peano y por lo tanto la lógica resultante es recursivamente axiomatizable, lo que no ocurre con DL.

Basándose en ideas semejantes a estas y en la lógica temporal de programas (confrontar (15) y (35)), Andréka, Németi y Sain presentaron su lógica NDL. Los programas y fórmulas son esencialmente los de DL, y las estructuras de NDL, definidas en 2.2, comprenden tres partes:

la del tiempo (que incluyen los distintos instantes de la ejecución de un programa), la de los datos (que son los objetos con los que trabaja el programa) y la de las sucesiones (que codifican las trazas de los programas: a cada instante le corresponde un valor de las variables). Al separar explicitamente el tiempo en las estructuras, se evitan de raiz los problemas de inexpresividad, sin que sea preciso como en el trabajo de Hajek , restringir la atención a modelos de la aritmética de Peano.

La intención es conseguir cálculos finitarios y completos en un sentido absoluto, es decir capaces de derivar formalmente todas las consecuencias de cualquier conjunto de axiomas. En 2.3 se define el cálculo NP y es probada su completitud. Ello puede considerarse como un primer objetivo de la lógica dinámica no standard:ofrecer un formalismo adecuado para razonar acerca de programas.

Un segundo objetivo consiste en emplear la lógica dinámica no standard para investigar las relaciones entre otras lógicas de programas. La sección 2.4 presenta algunos resultados en esta línea.

Añadiendo ciertos axiomas al cálculo NP, se rstringen las estructuras con las que se trabaja. Con esto podemos conseguir reforzar el sistema de inferencia. Esto es lo que se pretende con el conjunto de axiomas DAX.

Por otro lado, modificando dichos axiomas, podemos llegar a carac-

terizar dentro de NDL diversos sistemas de inferencia de otras lógicas. Un teorema "caracteriza" el cálculo P del capitulo 1 en un cierto sentido. La caracterización proporciona de hecho, un sistema formal equivalente a la lógica dinámica no standard de Hajek.

Finalmente la sección 2.5 presenta sumariamente la lógica de primer orden para programas recursivos de Cartwright (cfr. (6)). Esta lógica plantea otro ejemplo de semántica no standard como significado de programas, ahora funcionales y recursivos. Veremos como las ideas no standard permiten conciliar la semántica de punto fijo y la lógica de primer orden, pese a las conocidas críticas de Hitchcock y Park (21). La idea clave va a ser aquí la del mínimo punto fijo definible. Como se mostrará más tarde en el capítulo 3, esta noción introduce también, aunque de forma implícita, un tiempo de cómputo no standard.

2.2.- La lógica NDL. Definición.

Definición 2.2.1.- Tipo de similaridad td. Estructuras de tipo td

td es un tipo de similaridad de tres géneros t,d,s (llamados tiempo, datos y sucesiones respectivamente).

Los símbolos de td son:

- Los símbolos $\leqslant$, +, ·, 0, suc de la aritmética de Peano son del género t y forman el tipo de similaridad t del tiempo.
- Los símbolos de d, tipo de similaridad de los datos, son del género d
- ext es un símbolo de función con dos argumentos de género s y t y valor de género d

Las variables V_t= i,j,k.... , V_d= x,y,z... y V_s= u,v,w... son las correspondientes a los géneros t, d, s respectivamente.

De la forma habitual se definen los términos de t, d, s.

F_{td} son las fórmulas de primer orden heterogéneas de tipo td (donde $\tau \doteq \sigma$ solo se admite como fórmula si τ y σ son del mismo género)

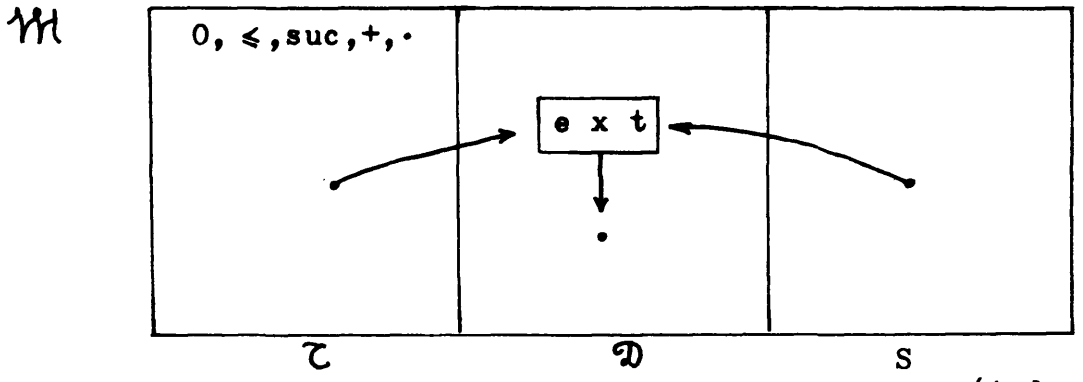
Una estructura del tipo td sera:

$$\mathcal{M} = (\mathcal{T}, \mathcal{D}, S, ext^{m})$$

donde $\mathcal{T}$ es una estructura del tipo de similaridad de la aritmética de Peano, $\mathcal{D}$ estructura del tipo de similaridad d, S conjunto no vacio y

$ext^{m}: S \times T \longrightarrow D$ (donde T es el universo de $\mathcal{T}$ y D es el universo de $\mathcal{D}$)

géneros	t tiempo	d datos	s sucesiones
símbolos	0, ≤, suc, +, ·	símbolos de d	
	e x t		
variables	i,j,k,...	x,y,z,...	u,v,w,...



Intuitivamente los elementos de S son sucesiones (indexadas en T) de elementos de D.

Como notación tomaremos:

F_d son las formulas de primer orden del tipo de similaridad d

T_d son los términos del tipo d (con las variables las de V_d)

Para $a,b \in T$ $\quad a < b \iff (a \leq b \wedge \neg a \doteq b)$

Definición 2.2.2.- Programas y fórmulas de NDL. PS_d y $NDFS_d$

PS_d

$\tau \in T_d,\ x \in V_d \Longrightarrow (x := \tau) \in PS_d$

$\varphi \in NDFS_d \Longrightarrow (\varphi?) \in PS_d$

$\alpha, \beta \in PS_d \Longrightarrow (\alpha ; \beta),\ (\alpha \cup \beta),\ (\alpha^*) \in PS_d$

$NDFS_d$

$\varphi \in F_{td} \Longrightarrow \varphi \in NDFS_d$

$\varphi, \psi \in NDFS_d$, $i \in V_t$, $x \in V_d$, $u \in V_s$ y $\alpha \in PS_d \Longrightarrow \neg\varphi, \varphi \wedge \psi, \exists i \varphi, \exists x \varphi,$

$\exists u \varphi, \langle\alpha\rangle\varphi \in NDFS_d$

Los programas PS_d de esta definición coinciden esencialmente con los de la definición 1.2.1, de aquí que conservemos su nombre. La construcción de las fórmulas $NDFS_d$ es tambien análoga a DFS_d. La principal diferencia está en que $NDFS_d$ incorpora características de la lógica temporal (cfr. (35)) a través de los géneros t y s.

Usaremos las abreviaturas de 1.2 $\forall, \vee, [\,]$.

Definición 2.2.3.- Estados sobre $\mathfrak{M}$. W_{td}

Sean $W_d = D^{V_d}$, $W_t = T^{V_t}$, $W_i = S^{V_s}$ (aplicaciones que asignan valores a las variables de cada género).

$$W_{td} = W_t \times W_d \times W_s$$

Definiremos, por inducción sobre la estructura, simultáneamente $\varphi^{\mathfrak{M}}$, significado de $\varphi \in NDFS_d$ en $\mathfrak{M}$, y $\alpha^{\mathfrak{M}}$, significado de $\alpha \in PS_d$ en $\mathfrak{M}$

$$\varphi^{\mathfrak{M}} : W_{td} \longrightarrow \{0,1\}$$

$$\alpha^{\mathfrak{M}} : W_{td} \times W_{td} \longrightarrow \{0,1\}$$

Como notación escribiremos:

$w \in W_{td}$ $\quad \mathfrak{M} \models_w \varphi \iff \varphi^{\mathfrak{M}}(w) = 1$

$w, w' \in W_{td}$ $\quad w \; \alpha^{\mathfrak{M}} \; w' \iff \alpha^{\mathfrak{M}}(w, w') = 1$

Definición 2.2.4.- Semántica de programas y fórmulas de NDL

Sea $w = (g, s, r) \in W_{td}$, $w' \in W_{td}$

$w \; (x := \tau)^{\mathfrak{M}} \; w' \iff w' = (g, s[\tau(s)^{\mathfrak{M}}/x], r)$

$w\,(\varphi?)^{\mathfrak{M}}\,w' \iff w = w'$ y $\mathfrak{M} \models_w \varphi$

$w\,(\alpha;\beta)^{\mathfrak{M}}\,w' \iff$ existe $w'' \in W_{td}$ con $w\ \alpha^{\mathfrak{M}} w''\ \beta^{\mathfrak{M}}\ w'$

$w\,(\alpha \cup \beta)^{\mathfrak{M}}\,w' \iff w\ \alpha^{\mathfrak{M}} w'$ o $w\ \beta^{\mathfrak{M}}\ w'$

Para definir $(\alpha^*)^{\mathfrak{M}}$ suponemos elegido $n \in N$ de forma que $\bar{x} = x_0,\ldots,x_n$ incluya todas las variables de género d que aparecen en α. Dados un estado $w = (g, s, r) \in W_{td}$ y una tupla $\bar{l} = l_0,\ldots,l_n$ de elementos de S, convenimos en que $\bar{l}_w(b)$ designa al estado:

$$(g,\ s[ext^{\mathfrak{M}}(l_0,b)/x_0,\ldots,ext^{\mathfrak{M}}(l_n,b)/x_n],\ r) \in W_{td}$$

para cada instante $b \in T$.

Decimos que $\bar{l}$ describe una traza de α en $\mathfrak{M}$ partiendo de w entre los instantes 0 y $a \in T$, y escribimos $Trz_{\alpha,w}(a,\bar{l})$ si y solo si se cumple

$\bar{l}_w(0^{\tau}) = w$ y

$\bar{l}_w(b)\ \alpha^{\mathfrak{M}}\ \bar{l}\ (suc^{\tau}(b))$ para todo $b \in T$, $b <^{\mathfrak{M}} a$

Finalmente, definimos

$w\,(\alpha^*)^{\mathfrak{M}}\,w' \iff$ existen $a \in T$, $\bar{l} \in S^{n+1}$ tal que

$$Trz_{\alpha,w}(a;\bar{l}) \quad y \quad \bar{l}_w(a) = w'$$

Como se vera más adelante esta noción de traza es definible en F_{td}

El resto de la semántica de NDL es como sigue

$\varphi \in F_{td}$ $\mathfrak{M} \models_w \varphi$ se define de la forma habitual

$\varphi, \psi \in NDFS_d$, $x \in V_d$, $i \in V_t$, $u \in V_s$

$$\mathfrak{M} \models_w \neg\varphi \iff \mathfrak{M} \not\models_w \varphi$$

$$\mathfrak{M} \models_w \varphi \wedge \psi \iff \mathfrak{M} \models_w \varphi \quad y \quad \mathfrak{M} \models_w \psi$$

$\mathfrak{M} \models_w \exists i\, \varphi \iff$ existe $a \in T$ con $\mathfrak{M} \models_{(g[a/i],s,r)} \varphi$

$\mathfrak{M} \models_w \exists x\, \varphi \iff$ existe $e \in D$ con $\mathfrak{M} \models_{(g,s[e/x],r)} \varphi$

$\mathfrak{M} \models_w \exists u\, \varphi \iff$ existe $l \in S$ con $\mathfrak{M} \models_{(g,s,r[l/u])} \varphi$

$\varphi \in NDFS_d$, $\alpha \in PS_d$

$\mathfrak{M} \models_w \langle \alpha \rangle \varphi \iff$ existe $w' \in W_{td}$ con $w\ \alpha^{\mathfrak{M}}\ w'$ y $\mathfrak{M} \models_{w'} \varphi$

Esta definición de semántica de fórmulas y programas es análoga a la definición 1.2.4. De hecho la única diferencia se halla en la definición de $(\alpha^*)^{\mathfrak{M}}$.

La "fórmula" $Trz_{\alpha, w}(a,\bar{l})$ indica que $\bar{l}$ es una "ejecución" del programa (α^*) "hasta" a (en el sentido de la relación $<^{\mathfrak{M}}$)

Notese que para cualquier $\alpha \in PS_d$ si $w = (g, s, r)$, $w' = (g', s', r')$ y $w\ \alpha^{\mathfrak{M}} w'$ tenemos $g = g'$ y $r = r'$. Es decir, la ejecución de cualquier programa no afecta a las partes g,r (de W_t, W_s) del estado.

Por la definición 2.2.4 tenemos:

$\mathfrak{M} \models_w [\alpha] \varphi \iff$ para todo $w' \in W_{td}$ con $w\ \alpha^{\mathfrak{M}}\ w'$ entonces $\mathfrak{M} \models_{w'} \varphi$

Ahora definimos

$\mathfrak{M} \models \varphi \iff$ Para todo $w \in W_{td}$ $\mathfrak{M} \models_w \varphi$

y se lee φ es válida en $\mathfrak{M}$.

Las definiciones de validez, consecuencia, etc. son las habituales (ver definición 1.2.5).

Así queda definida la lógica NDL, Lógica Dinámica no standard. El término no standard hace referencia por un lado a la semántica de la iteración $*$, que no es la habitual, y por otro lado indica que las estructuras del tiempo $\mathcal{T}$ y de los datos $\mathcal{D}$ pueden ser cualquier modelo de su tipo de similaridad.

En particular $\mathcal{T}$ puede ser cualquier modelo de la aritmética de Peano, en contraste con lo visto en el capítulo 1, donde la copia de N incluida en una estructura aritmética tenia que ser forzosamente standard

Pero además esta lógica permite trabajar en estructuras con dichas características. Si definimos dichas estructuras con una axiomatización $AX \subseteq NDFS_d$ podemos trabajar solo con modelos $\mathcal{M}$ que cumplan $\mathcal{M} \models AX$. Las trazas de los programas se comportan en $\mathcal{M}$ de manera más intuitiva, definiendo más claramente la semántica de los programas. En 2.4 discutiremos estos posibles axiomas con más detalle.

Aunque no entraremos en ello en el presente trabajo, también es posible en principio emplear el marco de NDL para estudiar estructuras de tiempo no secuencial, aptas, por ejemplo, para la programación paralela. Esta posibilidad ha sido sugerida por Andréka, Németi y Sain en (2).

2.3.- El cálculo NP. Teorema de completitud.

Notaciones y abreviaturas:

$\bar{x} = x_0,\ldots,x_n = var(\alpha)$ es el conjunto de variables (de género $\underline{d}$) que aparecen en el programa $\alpha \in PS_d$. A cada $\bar{x}$ asi definido asociamos un vector de variables de V_s de la misma dimensión $\bar{u} = u_0,\ldots,u_n$

Abreviaremos $\exists x_0 \exists x_1 \ldots \exists x_n$ por $\exists \bar{x}$. Analogamente abreviaremos $\forall \bar{x}$, $\exists \bar{u}$, $\forall \bar{u}$.

Además escribiremos $u_k(i)$ en lugar de $ext(u_k,i)$ y $\bar{u}(i)$ en lugar de $(u_0(i),\ldots,u_n(i))$.

La definición de sustitución $\varphi[\tau/x]$ ($\tau \in T_d$) es la dada en 1.3. Se sigue verificando un lema de sustitución:

Sea $w = (g,s,r) \in W_{td}$ $\quad \mathcal{M} \models_w \varphi[\tau/x] \iff \mathcal{M} \models_{(g,s[\tau(s)^{\mathcal{M}}/x],r)} \varphi$

La generalización de sustitución a vectores de variables $\bar{x}$ es la natural.

Definición 2.3.1.- Fórmula $\underline{traza}_{\alpha*}(i,\bar{u})$

$\underline{traza}_{\alpha*}(i,\bar{u}) = \forall j\ (j < i \longrightarrow \exists \bar{x}\ (\bar{x} \doteq \bar{u}(j) \wedge <\alpha> \bar{x} \doteq \bar{u}(suc(j))))$

$\underline{traza}_{\alpha*}(i,\bar{u}) \in NDFS_d$

Definición 2.3.2.- Cálculo NP

Axiomas

(T_0) Todas las instancias de tautologias del cálculo proposicional resultantes de sustituir consistentemente variables proposicionales por fórmulas de $NDFS_d$

(T_1) Un conjunto de axiomas para F_{td} que junto con las reglas (MP) y (G2) forme un sistema de inferencia completo para F_{td}

(:=) $\langle x:=\tau\rangle\varphi \longleftrightarrow \varphi[\tau/x] \qquad \tau\in T_d$

(?) $\langle\chi?\rangle\varphi \longleftrightarrow \chi \wedge \varphi$

(;) $\langle\alpha;\beta\rangle\varphi \longleftrightarrow \langle\alpha\rangle\langle\beta\rangle\varphi$

($\cup$) $\langle\alpha\cup\beta\rangle\varphi \longleftrightarrow \langle\alpha\rangle\varphi \vee \langle\beta\rangle\varphi$

(*) $\langle\alpha^*\rangle\varphi \longleftrightarrow \exists\bar{u}\,\exists i\,(\underline{\text{traza}}\ (i,\bar{u}) \wedge \bar{u}(0) \doteq \bar{x} \wedge \varphi[\bar{u}(i)/\bar{x}])$

$\bar{u}$ e i no aparecen en φ

Reglas

(MP) $$\frac{\varphi\ ,\ \varphi\longrightarrow\psi}{\psi}$$

(G1) $$\frac{\varphi\longrightarrow\psi}{\langle\alpha\rangle\varphi\longrightarrow\langle\alpha\rangle\psi}$$

(G2) $$\frac{\varphi\longrightarrow\psi}{\exists i\varphi\longrightarrow\exists i\psi}\ ,\ \frac{\varphi\longrightarrow\psi}{\exists x\varphi\longrightarrow\exists x\psi}\ ,\ \frac{\varphi\longrightarrow\psi}{\exists u\varphi\longrightarrow\exists u\psi}$$

Teorema 2.3.1.- Corrección de NP

Para cualquier $\Phi\subseteq NDFS_d$, $\varphi\in NDFS_d$ $\quad \Phi \overset{NP}{\vdash} \varphi \Rightarrow \Phi\models\varphi$

Demostración.-

Sea $\mathfrak{M}$ una estructura de tipo td con $\mathfrak{M}\models\Phi$. Basta probar la validez de los axiomas en $\mathfrak{M}$ y para las reglas, si las premisas son válidas en $\mathfrak{M}$, tambien lo es la conclusión.

La validez de los axiomas $(:=),(?),(;),(\cup)$ se demuestra de manera análoga al lema 1.3.2 y es consecuencia inmediata de la definicion de $\alpha'^{\mathfrak{M}}$. La validez de las reglas tambien es inmediata.

En cuanto a la validez en $\mathfrak{M}$ del axioma $(*)$ basta comprobar que la fórmula $\underline{traza}_{\alpha^*}(i,\bar{u})$ refleja trivialmente lo que hemos definido como semántica de la iteración.

Teorema 2.3.2.- Expresividad

Para cualquier $\varphi \in NDFS_d$ existe una fórmula $\varphi' \in F_{td}$ con las mismas variables libres que φ y tal que $\vdash^{NP} (\varphi \leftrightarrow \varphi')$

Demostración.-

Por inducción sobre la estructura de φ

$\varphi \in F_{td} \Longrightarrow \varphi' = \varphi$ y $(T_o) \vdash^{NP} (\varphi \leftrightarrow \varphi')$

$\varphi = \psi \wedge \chi \Longleftrightarrow$ por hipótesis de inducción $\vdash^{NP} (\psi \leftrightarrow \psi')$

$\vdash^{NP} (\chi \leftrightarrow \chi')$

con $\psi', \chi' \in F_{td}$. Tomamos $\varphi' = \psi' \wedge \chi'$ y $(T_o),(MP) \vdash^{NP} (\varphi \leftrightarrow \varphi')$

$\varphi = \exists i \psi$ o $\varphi = \exists x \psi$ o $\varphi = \exists u \psi$ como por hipótesis de inducción existe $\psi' \in F_{td}$ con $^{NP} (\psi \leftrightarrow \psi')$ tomando $\varphi' = \exists i \psi'$, $\varphi' = \exists x \psi'$, $\varphi' = \exists u \psi'$ según el caso

$(G2),(T_o),(MP) \vdash^{NP} (\varphi \leftrightarrow \varphi')$

$\varphi = \neg \psi$ y $\vdash^{NP} (\psi \leftrightarrow \psi')$ con $\psi' \in F_{td}$ por hipótesis de inducción, tomamos $\varphi' = \neg \psi'$ y $(T_o),(MP) \vdash^{NP} (\varphi \leftrightarrow \varphi')$

$\varphi = \langle \alpha \rangle \psi$ procedemos por inducción sobre la estructura de α .

Por hipótesis de inducción existe $\psi' \in F_{td}$ con $\vdash^{NP} (\psi \leftrightarrow \psi')$

$\alpha = (x := \tau)$

$(T_o),(MP),(G1) \quad \vdash^{NP} \quad \langle x := \tau \rangle \Psi \leftrightarrow \langle x := \tau \rangle \Psi'$

$(:=) \quad \vdash^{NP} \quad \langle x := \tau \rangle \Psi' \leftrightarrow \Psi'[\tau/x]$

$(MP),(T_o) \quad \vdash^{NP} \quad \langle x := \tau \rangle \Psi \leftrightarrow \Psi'[\tau/x]$

y obtenemos el resultado tomando $\varphi' = \Psi'[\tau/x]$

$\alpha = (\chi?)$

Por hipótesis de inducción existe $\chi' \in F_{td}$ con $\vdash^{NP} (\chi \leftrightarrow \chi')$

$(T_o),(MP),(G1) \quad \vdash^{NP} \quad \langle \chi? \rangle \Psi \leftrightarrow \langle \chi? \rangle \Psi'$

$(?) \quad \vdash^{NP} \quad \langle \chi? \rangle \Psi' \leftrightarrow (\chi \wedge \Psi')$

$(T_o),(MP) \quad \vdash^{NP} \quad (\chi \wedge \Psi') \leftrightarrow (\chi' \wedge \Psi')$

$(T_o),(MP) \quad \vdash^{NP} \quad \langle \chi? \rangle \Psi \leftrightarrow (\chi' \wedge \Psi')$

y obtenemos el resultado tomando $\varphi' = (\chi' \wedge \Psi')$

$\alpha = \beta \cup \beta'$

$(\cup) \quad \vdash^{NP} \quad \langle \beta \cup \beta' \rangle \Psi \leftrightarrow \langle \beta \rangle \Psi \vee \langle \beta' \rangle \Psi$

Por hipótesis de inducción existen Ψ'_1, $\Psi'_2 \in F_{td}$ con

$\vdash^{NP} \quad \langle \beta \rangle \Psi \leftrightarrow \Psi'_1$

$\vdash^{NP} \quad \langle \beta' \rangle \Psi \leftrightarrow \Psi'_2$

$(T_o),(MP) \quad \vdash^{NP} \quad \langle \beta \cup \beta' \rangle \Psi \leftrightarrow \Psi'_1 \vee \Psi'_2$

Obtenemos el resultado tomando $\varphi' = \Psi'_1 \vee \Psi'_2$

$\alpha = \beta ; \beta'$

$(;) \quad \vdash^{NP} \quad \langle \beta ; \beta' \rangle \Psi \leftrightarrow \langle \beta \rangle \langle \beta' \rangle \Psi$

Por hipótesis de inducción existe $\Psi'_1 \in F_{td}$ con

$\vdash^{NP} \quad \langle \beta' \rangle \Psi \leftrightarrow \Psi'_1$

$(T_o),(MP)$ $\vdash^{NP}$ $\langle\beta;\beta'\rangle\psi \longleftrightarrow \langle\beta\rangle\psi'_1$

Por hipótesis de inducción existe $\psi'_2 \in F_{td}$ con

$\vdash^{NP}$ $\langle\beta\rangle\psi'_1 \longleftrightarrow \psi'_2$

$(T_o),(MP)$ $\vdash^{NP}$ $\langle\beta;\beta'\rangle\psi \longleftrightarrow \psi'_2$

y el resultado se obtiene con $\varphi' = \psi'_2$

$\alpha = \beta^*$

$(T_o),(MP),(G1)$ $\vdash^{NP}$ $\langle\beta^*\rangle\psi \longleftrightarrow \langle\beta^*\rangle\psi'$

$(*)$ $\vdash^{NP}$ $\langle\beta^*\rangle\psi' \longleftrightarrow \exists\bar{u}\,\exists i\,(\underline{traza}_{\beta^*}(i,\bar{u}) \wedge \bar{u}(0) \doteq \bar{x} \wedge \psi'[\bar{u}(i)/\bar{x}])$

Por hipótesis de inducción existe $\psi'_1 \in F_{td}$ con

$\vdash^{NP}$ $\langle\beta\rangle(\bar{x} \doteq \bar{u}(suc(j))) \longleftrightarrow \psi'_1$

$(T_o),(MP),(G2)$ $\vdash^{NP}$ $\underline{traza}_{\beta^*}(i,\bar{u}) \longleftrightarrow \psi'_2$

con $\psi'_2 = \forall j\,(j<i \longrightarrow \exists\bar{x}\,(\bar{x} \doteq \bar{u}(j) \wedge \psi'_1))$

$(T_o),(MP)$ $\vdash^{NP}$ $\langle\beta^*\rangle\psi \longleftrightarrow \psi'_3$

con $\psi'_3 = \exists\bar{u}\,\exists i\,(\psi'_2 \wedge \bar{u}(0) \doteq \bar{x} \wedge \psi'[\bar{u}(i)/\bar{x}])$ y $\psi'_3 \in F_{td}$,

luego con $\varphi' = \psi'_3$ obtenemos el teorema.

Teorema 2.3.3.- Completitud de NP

Para cualquier $\Phi \subseteq NDFS_d$, $\varphi \in NDFS_d$

$$\Phi \models \varphi \Longrightarrow \Phi \vdash^{NP} \varphi$$

Demostración.-

Por el teorema 2.3.2 existe $\varphi' \in F_{td}$ con

$\vdash^{NP}$ $(\varphi \longleftrightarrow \varphi')$

Como $\Phi \models \varphi$ por corrección de NP $\Phi \models \varphi'$

Ahora con $(T_1),(MP),(G2)$ $\Phi \vdash^{NP} \varphi'$ pues $\varphi' \in F_{td}$

y $(T_o),(MP)$ $\Phi \vdash^{NP} \varphi$

El cálculo NP es similar al cálculo P (ver definicion 1.6.1) para DL, reemplazando las reglas (I) y (C) por el axioma (*) de NDL.

El axioma (*) describe la sémantica de la iteración mediante una fórmula de $NDFS_d$, que solo hace referencia a α . Ello hace posible la prueba del teorema 2.3.2, que puede entenderse como un resultado de expresividad similar al teorema 1.5.2, decisivo en la demostración del teorema de completitud de Harel.

Por supuesto, la expresividad que ahora conseguimos se debe a que la semántica de la iteración no es la standard.

2.4.- Potencia de la lógica NDL. Comparación con otras lógicas dinámicas.

Como vimos en 1.4 el cálculo P es completo con respecto a cualquier estructura aritmética $\mathfrak{D}$. Puesto que la teoría de primer orden de una estructura aritmética es altamente indecidible, en la práctica se trabajará con una subteoría recursivamente axiomatizable de $Th(\mathfrak{D})$. Resulta natural imaginar que esa subteoría contenga los axiomas de Peano.

De la misma forma se pueden añadir a NDL ciertos axiomas complementarios para reforzar el sistema. Esto nos permitira primero trabajar con modelos que cumplan una cierta especificación.

Los nuevos axiomas postularán propiedades acerca de la estructu del tiempo $\mathcal{T}$ y del conjunto de sucesiones S, tendiendo a que expresen nuestras ideas intuitivas acerca del tiempo y de las trazas.

En segundo lugar estos axiomas nos permitirán comparar la potencia deductiva de NDL con otros sistemas. En particular caracterizaremos el cálculo P dentro de NDL en un cierto sentido.

Definicion 2.4.1.- SDF_d, SPS_d

SPS_d

$x \in V_d$, $\tau \in T_d \Longrightarrow (x := \tau) \in SPS_d$

$\varphi \in SDF_d \Longrightarrow (\varphi ?) \in SPS_d$

$\alpha, \beta \in SPS_d \Longrightarrow (\alpha;\beta), (\alpha \cup \beta), (\alpha^*) \in SPS_d$

SDF_d

$F_d \subset SDF_d$

$\varphi, \psi \in SDF_d, x \in V_d \Longrightarrow \neg\varphi, \varphi \wedge \psi, \exists x \varphi \in SDF_d$

$\alpha \in SPS_d, \varphi \in SDF_d \Longrightarrow <\alpha>\varphi \in SDF_d$

La diferencia entre $NDFS_d$ y SDF_d está en que $NDFS_d$ está basado en F_{td} y SDF_d está basado en F_d. SDF_d son las fórmulas que solo hablan de los datos. De hecho SDF_d puede considerarse igual al conjunto DFS_d de la definición 1.2.1.

Definición 2.4.2.-

PA_t son los axiomas de Peano para el género t.

$TIA_{td} = \{ \underline{ind}_t(\varphi) : \varphi \in F_{td} \}$

con $\underline{ind}_t(\varphi) = ((\varphi[0/i] \wedge \forall i(\varphi \longrightarrow \varphi[suc(i)/i])) \longrightarrow \forall i \varphi)$

$DIA_{td} = \{ \underline{ind}_d(\varphi) : \varphi \in F_{td} \}$

con $\underline{ind}_d(\varphi) = ((\varphi[0'/x] \wedge \forall x(\varphi \longrightarrow \varphi[suc'(x)/x])) \longrightarrow \forall x \varphi)$

suponiendo que $0', suc' \in d$

PA_d son los axiomas de Peano para el género d, supuesto que $0', suc', \leq', +', \cdot' \in d$

$DIA_d = \{ \underline{ind}_d(\varphi) : \varphi \in F_d \}$ suponiendo que $0', suc' \in d$

$EXT = \{ \underline{ext}(\varphi) : \varphi \in F_{td} \}$

con $\underline{ext}(\varphi) = \forall j \exists x \varphi \longrightarrow \exists u \forall j \varphi[u(j)/x]$

$EXT^b = \{ \underline{ext}^b(\varphi) : \varphi \in F_{td} \}$

con $\underline{ext}^b(\varphi) = \forall j(j \leq i \longrightarrow \exists!x\,\varphi) \longrightarrow \exists u \forall j(j \leq i \longrightarrow \varphi[u(j)/x])$

Además podemos considerar el cálculo P (ver definicion 1.6.1) enunciado en NDL sustituyendo (T∃) por (T_o) y (T_1) definiendo:

(T_1) Un conjunto de axiomas para F_d que junto con las reglas (MP) y (G2) forme un sistema de inferencia completo para F_d

Para no complicar la notación lo continuaremos llamando P.

Definición 2.4.3.-

Sean $\vdash^1$, $\vdash^2$ dos sistemas de inferencia en NDL, y sean AX1, AX2 $\subseteq NDFS_d$

$(AX1 \vdash^1) \underset{SDF}{\leq} (AX2 \vdash^2)$ $\iff$ Para cualquier $\Phi \subseteq SDF_d$, $\varphi \in SDF_d$

$\Phi \cup AX1 \vdash^1 \varphi \implies \Phi \cup AX2 \vdash^2 \varphi$

$(AX1 \vdash^1) \underset{SDF}{<} (AX2 \vdash^2)$ $\iff$ $(AX1 \vdash^1) \underset{SDF}{\leq} (AX2 \vdash^2)$ y no $(AX2 \vdash^2) \underset{SDF}{\leq} (AX1 \vdash^1)$

$(AX1 \vdash^1) \underset{SDF}{\equiv} (AX2 \vdash^2)$ $\iff$ $(AX1 \vdash^1) \underset{SDF}{\leq} (AX2 \vdash^2)$ y $(AX2 \vdash^2) \underset{SDF}{\leq} (AX1 \vdash^1)$

Estas definiciones nos van a permitir comparar sistemas de inferencia a partir de ciertos conjuntos de axiomas. En particular nos va a permitir comparar NP y P

Teorema 2.4.1.-

Sea d un tipo de similaridad que contiene una copia disjunta del tipo de similaridad de la aritmética de Peano. Entonces:

$(PA_d \cup DIA_d \vdash^P) \underset{SDF}{\equiv} (PA_t \cup TIA_{td} \cup PA_d \cup DIA_{td} \cup EXT^b \vdash^{NP})$

Demostración.-

En lo que sigue llamaremos AX1= $PA_d \cup DIA_d$, AX2= $PA_t \cup TIA_{td} \cup PA_d \cup DIA_{td} \cup EXT^b$.

($\underset{SDF}{\leq}$)

Para cualquier $\varphi \in SDF_d$, $\Phi \subseteq SDF_d$

$$AX1 \cup \Phi \vdash^{P} \varphi \implies AX2 \cup \Phi \vdash^{NP} \varphi$$

Sea $\mathcal{M} \models AX2 \cup \Phi$

Si probamos que $\mathcal{M} \models \varphi$ entonces por completitud del cálculo NP tendremos $AX2 \cup \Phi \vdash^{NP} \varphi$

Sea $\varphi_0, \varphi_1, \ldots, \varphi_n = \varphi$ la $\vdash^{P}$ demostración de φ

Probemos por inducción que $\mathcal{M} \models \varphi_i$ para cualquier i $0 \leq i \leq n$

Para i, $0 \leq i \leq n$ φ_i es un axioma de P o $\varphi_i \in \Phi \cup AX1$ o φ_i se obtiene por aplicación de una regla de P

Por el teorema de corrección (teorema 2.3.1) todos los axiomas de P son validos en $\mathcal{M}$.

Si $\varphi_i \in \Phi \cup AX1$ tambien podemos asegurar que $\mathcal{M} \models \varphi_i$ (pues $DIA_d \subseteq DIA_{td}$)

Además las reglas (MP), (G1) y (G2) tambien son válidas, por corrección, en $\mathcal{M}$.

Queda probar que para las reglas (I) y (C) si las premisas son válidas en $\mathcal{M}$, también lo son las conclusiones.

(I) $$\frac{\varphi \longrightarrow [\alpha]\varphi}{\varphi \longrightarrow [\alpha^*]\varphi}$$

Supongamos $\mathcal{M} \models \varphi \longrightarrow [\alpha]\varphi$

Sea $w = (g, s, r) \in W_{td}$ y supongamos $\mathcal{M} \models_w \varphi$

Hemos de probar

$$\mathcal{M} \models_w [\alpha^*]\varphi \iff \mathcal{M} \models_w \forall i \, \forall \bar{u} \, (\underline{traza}_{\alpha^*}(i,\bar{u}) \wedge \bar{u}(0) \doteq \bar{x} \longrightarrow \varphi[\bar{u}(i)/\bar{x}])$$

con $\bar{x} = var(\alpha)$

Consideremos la fórmula

$\psi = \forall \bar{u}(\underline{traza}_{\alpha^*}(i,\bar{u}) \wedge \bar{u}(0) \doteq \bar{x} \longrightarrow \varphi[\bar{u}(i)/\bar{x}])$ que tiene a i entre sus variables libres. Por el teorema 2.3.2 existe $\psi' \in F_{td}$ con i entre sus variables libres tal que $\mathcal{M} \models \psi \longleftrightarrow \psi'$

Para probar $\mathcal{M} \models_w [\alpha^*]\varphi$ aplicaremos el axioma de inducción de $TIA_{td}, \underline{ind}_t$ a $\psi'(i)$

· $\psi'[0/i]$

Como $\mathcal{M} \models_w \varphi$ tenemos $\mathcal{M} \models_w \psi[0/i]$ y por tanto $\mathcal{M} \models_w \psi'[0/i]$

· $\psi' \longrightarrow \psi'[suc(i)/i]$

Supongamos $\mathcal{M} \models_w \psi'$ y para $\bar{l} \in S^{n+1}$

$$\mathcal{M} \models_w \underline{traza}_{\alpha^*}(suc(i),\bar{l}) \wedge \bar{l}(0) \doteq \bar{x}$$

$\mathcal{M} \models_w \psi'$ supone $\mathcal{M} \models_w \psi$. Además como $\mathcal{M} \models PA_t$ la relación $<^{\mathcal{M}}$ es transitiva y $\mathcal{M} \models_w \underline{traza}_{\alpha^*}(suc(i),\bar{l}) \longrightarrow \underline{traza}_{\alpha^*}(i,\bar{l})$

Por tanto $\mathcal{M} \models_w \varphi[\bar{l}(i)/\bar{x}]$ y por el lema de sustitución

$$\mathcal{M} \models_{(g,s[\bar{l}(i)^{\mathcal{M}}/\bar{x}],r)} \varphi$$

Como $\mathcal{M} \models \varphi \longrightarrow [\alpha]\varphi$ tenemos $\mathcal{M} \models_{(g,s[\bar{l}(i)^{\mathcal{M}}/\bar{x}],r)} [\alpha]\varphi$

Por otra parte $\mathcal{M} \models_w \underline{traza}_{\alpha^*}(suc(i),\bar{l})$

Como $\mathcal{M} \models PA_t$ se cumple $\mathcal{M} \models i < suc(i)$ y tenemos

$$\mathcal{M} \underset{w}{\models} \exists \bar{x}(\bar{x} \doteq \bar{l}(i) \wedge \langle \alpha \rangle (\bar{x} \doteq \bar{l}(suc(i)))$$

$$\Longrightarrow \mathcal{M} \underset{(g, s[\bar{l}(i)^{\mathcal{M}}/\bar{x}], r)}{\models} \langle \alpha \rangle (\bar{x} \doteq \bar{l}(suc(i)))$$

$\Longrightarrow$ existe $w' = (g, t, r)$ con

$(g, s[\bar{l}(i)^{\mathcal{M}}/\bar{x}], r) \; \alpha^{\mathcal{M}} \; w'$ y $\mathcal{M} \underset{w'}{\models} \bar{x} \doteq \bar{l}(suc(i))$

$\Longrightarrow$ ($\bar{x} = var(\alpha)$ y α no afecta a las otras variables)

$(g, s[\bar{l}(i)^{\mathcal{M}}/\bar{x}], r) \; \alpha^{\mathcal{M}} \; (g, s[\bar{l}(suc(i))^{\mathcal{M}}/\bar{x}], r)$

Como $\mathcal{M} \underset{(g, s[\bar{l}(i)^{\mathcal{M}}/\bar{x}], r)}{\models} [\alpha] \varphi \Longrightarrow$ para todo $w' \in W_{td}$ con

$(g, s[\bar{l}(i)^{\mathcal{M}}/\bar{x}], r) \; \alpha^{\mathcal{M}} \; w'$ entonces $\mathcal{M} \underset{w'}{\models} \varphi$

Luego $\mathcal{M} \underset{(g, s[\bar{l}(suc(i))^{\mathcal{M}}/\bar{x}], r)}{\models} \varphi$ y por el lema de sustitución

$\mathcal{M} \underset{w}{\models} \varphi [\bar{l}(suc(i))/\bar{x}]$

Por tanto $\mathcal{M} \underset{w}{\models} \psi \longrightarrow \psi [suc(i)/\bar{x}]$ y

$\mathcal{M} \underset{w}{\models} \psi' \longrightarrow \psi' [suc(i)/\bar{x}]$

Con el axioma de inducción $\underline{ind}_t(\psi')$ concluimos $\mathcal{M} \models \forall i \, \psi'$

y como consecuencia $\mathcal{M} \models [\alpha^*] \varphi$

(C) $$\frac{\varphi [suc'(x)/x] \longrightarrow \langle \alpha \rangle \varphi}{\varphi \longrightarrow \langle \alpha^* \rangle \varphi [0'/x]}$$ x no aparece en α

Supongamos $\mathcal{M} \models \varphi [suc'(x)/x] \longrightarrow \langle \alpha \rangle \varphi$

Sea $w = (g, s, r) \in W_{td}$ y supongamos $\mathcal{M} \underset{w}{\models} \varphi$

Hemos de probar que $\mathcal{M} \underset{w}{\models} \langle \alpha^* \rangle \varphi [0'/x]$

Por el teorema 2.3.2 existe $\psi' \in F_{td}$ con

$$\mathcal{M} \models \langle \alpha^* \rangle \varphi[0'/x] \longleftrightarrow \psi'$$

Demostraremos $\mathcal{M} \underset{w}{\models} \langle \alpha^* \rangle \varphi[0'/x]$ por inducción sobre $s(x) = m \in D$ y simultaneamente para todo S, usando el esquema de inducción de DIA_{td}: $\underline{ind}_d$ aplicado a ψ'

Supondremos $\bar{x} = var(\alpha)$ y que x no está en $\bar{x}$

$m = 0'$

Como $\mathcal{M} \models EXT^b$ tenemos

$\mathcal{M} \underset{w}{\models} \underline{ext}^b(y \doteq s(x_o)), \ldots\ldots, \mathcal{M} \underset{w}{\models} \underline{ext}^b(y \doteq s(x_n))$ y dado que

$\mathcal{M} \underset{w}{\models} \exists! y(y \doteq s(x_o)), \ldots\ldots, \mathcal{M} \underset{w}{\models} \exists! y(y \doteq s(x_n))$ tenemos

$\mathcal{M} \underset{w}{\models} \exists u_o \forall j(j \leq 0 \longrightarrow u_o(j) \doteq s(x_o)), \ldots\ldots\ldots,$

$\mathcal{M} \underset{w}{\models} \exists u_n \; j(j \leq 0 \longrightarrow u_n(j) \doteq s(x_n))$ y claramente

$$\mathcal{M} \underset{w}{\models} \exists \bar{u} \; \underline{traza}_{\alpha^*}(0, \bar{u})$$

Como $\mathcal{M} \underset{w}{\models} \varphi$ y $s(x) = 0'$ tenemos $\mathcal{M} \underset{w}{\models} \varphi[0'/x]$ y por tanto

$\mathcal{M} \underset{w}{\models} \exists \bar{u}(\underline{traza}_{\alpha^*}(0, \bar{u}) \wedge \bar{u}(0) \doteq \bar{x} \wedge \varphi[0'/x])$

Luego $\mathcal{M} \underset{w}{\models} \langle \alpha^* \rangle \varphi[0'/x]$ y $\mathcal{M} \underset{w}{\models} \psi'$

$m' = suc'(m)$

Consideremos $s_1 = s[m/x]$ y $w' = (g, s_1, r)$

$$\mathcal{M} \underset{w}{\models} \varphi \Longrightarrow \mathcal{M} \underset{w'}{\models} \varphi[suc'(x)/x]$$

Como $\mathcal{M} \models \varphi[suc'(x)/x] \longrightarrow \langle \alpha \rangle \varphi$ por hipótesis tenemos

$\mathcal{M} \underset{w'}{\models} \langle \alpha \rangle \varphi$, luego existe $w'' = (g, t, r)$ con $w' \; \alpha^{\mathcal{M}} \; w''$ y

$\mathcal{M} \underset{w''}{\models} \varphi$

Dado que x no aparece en α tenemos $w \; \alpha^{\mathcal{M}} w''$ y además $t(x) = s_1(x) = m$

Podemos aplicar la hipótesis de inducción al estado w" y obtenemos

$$\mathcal{M} \models_{w''} \langle \alpha^* \rangle \varphi [0'/x]$$

Sean $\bar{l} \in S^{n+1}$, $a \in T$ con

$$\mathcal{M} \models_{w''} \underline{traza}_{\alpha^*}(a,\bar{l}) \wedge \bar{l}(0) \doteq \bar{x} \wedge \varphi[\bar{l}(a)/\bar{x}, 0'/x]$$

Para cada k, $0 \leq k \leq n$ consideramos la fórmula (con parametros)

$$\varphi_k = ((j \doteq 0 \longrightarrow y \doteq s(x_k)) \wedge (\neg j \doteq 0 \longrightarrow \exists j'(suc(j') \doteq j \wedge x \doteq l_k(j'))))$$

Como $\mathcal{M} \models EXT^b$ para cada k tenemos $\mathcal{M} \models \underline{ext}^b(\varphi_k)$

Además $\mathcal{M} \models \forall j(j \leq a \longrightarrow \exists! x \varphi_k)$, luego existen $l'_0, \ldots, l'_n = \bar{l}' \in S^{n+1}$

con $\mathcal{M} \models \bar{l}'(0) \doteq s(\bar{x}) \wedge \forall j(j \leq a \longrightarrow \bar{l}'(suc(j)) \doteq \bar{l}(j))$

usando también $\mathcal{M} \models PA_t$

De $\mathcal{M} \models_{w} \underline{traza}_{\alpha^*}(a,\bar{l})$ tenemos que

$$\mathcal{M} \models_{w} \forall j(suc(0) \leq j < suc(a) \longrightarrow \exists \bar{x}(\bar{x} \doteq \bar{l}'(j) \wedge \langle \alpha \rangle (\bar{x} \doteq \bar{l}'(suc(j)))))$$

Además $\mathcal{M} \models_{w''} \bar{l}(0) \doteq \bar{x}$ y $w \alpha^{\mathcal{M}} w''$ por tanto

$$\mathcal{M} \models_{w} \exists \bar{x}(\bar{x} \doteq \bar{l}'(0) \wedge \langle \alpha \rangle (\bar{x} \doteq \bar{l}'(suc(0))))$$

Luego $\mathcal{M} \models_{w} \underline{traza}_{\alpha^*}(suc(a),\bar{l}') \wedge \bar{l}'(0) \doteq \bar{x}$

Como $\mathcal{M} \models_{w''} \varphi[\bar{l}(a)/\bar{x}, 0'/x]$ se tiene

$$\mathcal{M} \models_{w} \varphi[\bar{l}'(suc(a))/\bar{x}, 0'/x]$$

y obtenemos

$$\mathcal{M} \models_{w} \langle \alpha^* \rangle \varphi [0'/x]$$

y por tanto

$$\mathcal{M} \models_{w} \psi'$$

Con $\underline{ind}_d(\psi')$ concluimos $\mathcal{M} \models \langle \alpha^* \rangle \varphi [0'/x]$

($\overset{\geq}{\text{SDF}}$)

Para cualquier $\Phi \subseteq SDF_d$ y $\varphi \in SDF_d$

$$AX2 \cup \Phi \overset{NP}{\vdash} \varphi \Rightarrow AX1 \cup \Phi \overset{P}{\vdash} \varphi$$

Definición 2.4.4.- Modelo de trazas internas

Sea d un tipo de similaridad que incluye una copia disjunta de los símbolos de la aritmética de Peano ($\{0', suc', +', \cdot', \leq'\} \subset d$) y $\mathfrak{D}$ estructura de tipo d tal que $\mathfrak{D} \models PA_d$. Se define el modelo de trazas internas: $\mathcal{M}_{\mathfrak{D}} = (\mathcal{T}_{\mathfrak{D}}, \mathfrak{D}, D, ext^{\mathcal{M}_{\mathfrak{D}}})$ con D universo de $\mathfrak{D}$, $\mathcal{T}_{\mathfrak{D}}$ se obtiene tomando como universo D e interpretando los símbolos de t en $\mathcal{T}_{\mathfrak{D}}$ con las interpretaciones de $0', suc', +', \cdot', \leq'$ en $\mathfrak{D}$, y $ext^{\mathcal{M}_{\mathfrak{D}}} : D \times D \longrightarrow D$ con $ext^{\mathcal{M}_{\mathfrak{D}}} = comp^{\mathfrak{D}}$ donde $comp^{\mathfrak{D}}$ es la función vista en la definición 1.5.1 según la función beta de Gödel.

Veamos, previamente a la demostración, dos lemas sobre el comportamiento de $\mathcal{M}_{\mathfrak{D}}$ y un lema de expresividad para AX1.

Lema 2.4.1.-

Para cada fórmula $\varphi = \varphi(\overset{k}{i}, \overset{p}{x}, \overset{q}{u}) \in DFS_d$ (donde $\overset{k}{i}, \overset{p}{x}, \overset{q}{u}$ indican las variables libres de género t, d y s respectivamente) puede construirse una fórmula $\varphi^d = \varphi^d(\overset{k}{z}, \overset{p}{x}, \overset{q}{y}) \in F_d$, que verifica:

Si $\mathfrak{D} \models PA_d$ y $\overset{k}{m}, \overset{p}{n}, \overset{q}{l} \in D$ entonces:

$$\mathcal{M}_{\mathfrak{D}} \models \varphi[\overset{k}{m}/\overset{k}{z}, \overset{p}{n}/\overset{p}{x}, \overset{q}{l}/\overset{q}{u}] \iff \mathfrak{D} \models \varphi^d[\overset{k}{m}/\overset{k}{z}, \overset{p}{n}/\overset{p}{x}, \overset{q}{l}/\overset{q}{y}]$$

Demostración.-

Por el teorema 2.3.2 sabemos que para cada $\varphi \in DFS_d$ puede construir

se $\varphi' \in F_{td}$ con las mismas variables libres que φ y equivalente a φ en cualquier $\mathcal{M}$. Basta pues construir φ^d para $\varphi \in F_{td}$ y poner $\varphi^d = (\varphi')^d$ para las restantes fórmulas.

Sea X el conjunto de las infinitas variables de V_d que no aparezcan (ni libres ni ligadas) en φ. Por cada variable i de V_d que aparezca (libre o ligada) en φ elegimos una variable z de X. Análogamente, por cada variable $u \in V_s$ escojemos una variable y de X. De esta forma quedan definidas en particular z^k, y^q.

Sea $\underline{beta}(x_1, x_2, x_3)$ la fórmula de F_d que define la función beta de Gödel en PA_d. Para construir φ^d reemplazamos cada subfórmula de φ del tipo $ext(u, \sigma) \doteq \tau$ por $\underline{beta}(y, \sigma, \tau)$ que abreviaremos por $y(\sigma) \doteq \tau$.

Reemplazaremos además cada aparición de i por z y cada cuantificación de u por una cuantificación de y. Es obvio que esta construcción tiene las propiedades deseadas.

Lema 2.4.2.-

Sea d tipo de similaridad que incluye una copia disjunta de los símbolos de la aritmética de Peano, y $\mathfrak{D}$ estructura de tipo d

$$\mathfrak{D} \models AX1 \implies \mathcal{M}_{\mathfrak{D}} \models AX2$$

Demostración.-

1.- $\mathcal{M}_{\mathfrak{D}} \models PA_d$ ya que $\mathfrak{D} \models PA_d$

2.- $\mathcal{M}_{\mathfrak{D}} \models PA_t$ pues $\mathfrak{D} \models PA_d$ y $\mathcal{T}_{\mathfrak{D}}$ es una "copia restringida a t" de $\mathfrak{D}$

3.- $\mathcal{M}_{\mathfrak{D}} \models \mathrm{DIA}_{td}$

Dada $\varphi \in F_{td}$, el que $\mathcal{M}_{\mathfrak{D}} \models \underline{ind}_d(\varphi)$ es consecuencia de que $\mathfrak{D} \models \underline{ind}_d(\varphi^d)$, pues $\mathfrak{D} \models \mathrm{DIA}_d$, y del lema 2.4.1.

4.- $\mathcal{M}_{\mathfrak{D}} \models \mathrm{TIA}_{td}$ análogo a 3

5.- $\mathcal{M}_{\mathfrak{D}} \models \mathrm{EXT}^b$

Cada axioma de EXT^b es de la forma

$$ext^b(\varphi) = \forall j(j \leq i \longrightarrow \exists! x \varphi) \longrightarrow \exists u \forall j(j \leq i \longrightarrow \varphi[u(j)/x])$$

Según el lema 2.4.1 basta demostrar que $\mathfrak{D} \models (ext^b(\varphi))^d$

$$(ext^b(\varphi))^d = \forall z(z \leq' z_1 \longrightarrow \exists! x \varphi^d) \longrightarrow \exists y \forall z(z \leq' z_1 \longrightarrow \varphi^d[y(z)/x])$$

Ahora bien, suponiendo que $\mathfrak{D} \models \forall z(z \leq' z_1 \longrightarrow \exists! x \varphi^d)$ la existencia de y, tal que $\mathfrak{D} \models \forall z(z \leq' z_1 \longrightarrow \varphi^d[y(z)/x])$ se puede demostrar por inducción sobre z_1 (inducción válida pues $\mathfrak{D} \models \mathrm{DIA}_d$) utilizando propiedades de la función beta que son demostrables a partir de los axiomas de Peano (válidos en $\mathfrak{D}$). Más en concreto, la inducción sobre z_1 es informalmente:

$z_1 = 0'$ Sea x_o el único x de D tal que $\mathfrak{D} \models \varphi^d[0'/z_1, x_o/x]$.
En D hay un elemento y_o que cumple $comp^{\mathfrak{D}}(y_o, 0') = x_o$ y este es el buscado.

$z_1 = suc'(n)$ Por hipótesis de inducción existe y en D que verifica $\mathfrak{D} \models \forall z(z \leq' n \longrightarrow \varphi^d[(y(z)/x])$
Además sabemos que $\mathfrak{D} \models \exists! x \varphi^d[suc'(n)/z_1]$
Sea x_{n+1} el único x tal que $\mathfrak{D} \models \varphi^d[suc'(n)/z_1, x_{n+1}/x]$ y sea $y' \in D$ tal que $comp^{\mathfrak{D}}(y', z) = comp^{\mathfrak{D}}(y, z)$ para $z \leq n$ y $comp^{\mathfrak{D}}(y', suc'(n)) = x_{n+1}$
Entonces y' cumple lo deseado.

Lema 2.4.3.-

Para cualquier $\varphi \in SDF_d$ $\quad PA_d \overset{P}{\vdash} (\varphi \leftrightarrow \varphi^d)$

Demostración.-

Procedemos por inducción sobre la estructura de φ

$\varphi \in F_d \Longrightarrow \varphi^d = \varphi$ y (T_o) $AX1 \overset{P}{\vdash} (\varphi \leftrightarrow \varphi^d)$

$\varphi = \neg\psi \Longrightarrow \varphi^d = \neg\psi^d$ y por hipótesis de inducción

$AX1 \overset{P}{\vdash} \psi \leftrightarrow \psi^d$, luego $(T_o),(MP)$ $AX1 \overset{P}{\vdash} \neg\psi \leftrightarrow \neg\psi^d$

$\varphi = \exists x\,\psi \Longrightarrow \varphi^d = \exists x\,\psi^d$ y por hipótesis de inducción

$AX1 \overset{P}{\vdash} \psi \leftrightarrow \psi^d$, y $(T_o),(MP),(G2)$ $AX1 \overset{P}{\vdash} \exists x\,\psi \leftrightarrow \exists x\,\psi^d$

$\varphi = \psi \wedge \chi \Longrightarrow \varphi^d = \psi^d \wedge \chi^d$ y por hipótesis de inducción

$AX1 \overset{P}{\vdash} \psi \leftrightarrow \psi^d$

luego

$AX1 \overset{P}{\vdash} \chi \leftrightarrow \chi^d$

$(T_o),(MP)$ $AX1 \overset{P}{\vdash} \psi \wedge \chi \leftrightarrow \psi^d \wedge \chi^d$

$\varphi = \langle\alpha\rangle\psi$

Probaremos el resultado por inducción sobre α

Además probaremos simultáneamente, y como lema auxiliar que:

(b) para todo $\alpha \in PS_d$, $\chi \in F_d$, $\bar{x} = var(\alpha)$

$$AX1 \overset{P}{\vdash} \langle\alpha\rangle\chi \leftrightarrow \exists\bar{x}'(\chi[\bar{x}'/\bar{x}] \wedge \langle\alpha\rangle\bar{x} \doteq \bar{x}')$$

· $\alpha = (x := \tau)$

(a) $\varphi^d = (\psi[\tau/x])^d = \psi^d[\tau/x]$

Por hipótesis de inducción $AX1 \overset{P}{\vdash} \psi \leftrightarrow \psi^d$

$(T_1),(MP),(G2)$ $AX1 \overset{P}{\vdash} \psi[\tau/x] \leftrightarrow \psi^d[\tau/x]$

$(:=)$ $AX1 \overset{P}{\vdash} \varphi \leftrightarrow \psi[\tau/x]$

$(T_o),(MP)$ $\quad AX1 \overset{P}{\vdash} \varphi \leftrightarrow \psi^d[\tau/x]$

$AX1 \overset{P}{\vdash} \varphi \leftrightarrow \varphi^d$

(b)

$(:=)$ $\quad AX1 \overset{P}{\vdash} \langle x:=\tau\rangle\chi \leftrightarrow \chi[\tau/x]$

$(T_1),(MP),(G2)$ $\quad AX1 \overset{P}{\vdash} \chi[\tau/x] \leftrightarrow \exists x'(\chi[x'/x] \wedge x' \doteq \tau)$

$(:=)$ $\quad AX1 \overset{P}{\vdash} \langle x:=\tau\rangle x \doteq x' \leftrightarrow x' \doteq \tau$

$(T_o),(MP)$ $\quad AX1 \overset{P}{\vdash} \langle x:=\tau\rangle\chi \leftrightarrow \exists x'(\chi[x'/x] \wedge \langle x:=\tau\rangle x \doteq x')$

• $\alpha = (\psi_1?)$

(a) $\varphi^d = (\psi_1 \wedge \psi)^d = \psi_1^d \wedge \psi^d$

Con la hipótesis de inducción aplicada a ψ_1 y a ψ y (T_o), (MP)

tenemos $\quad AX1 \overset{P}{\vdash} \varphi^d \leftrightarrow \psi_1 \wedge \psi$

$(?)$ $\quad AX1 \overset{P}{\vdash} \psi_1 \wedge \psi \leftrightarrow \langle\psi_1?\rangle\psi$

$(T_o),(MP)$ $\quad AX1 \overset{P}{\vdash} \varphi^d \leftrightarrow \varphi$

(b)

$(?)$ $\quad AX1 \overset{P}{\vdash} \langle\psi_1?\rangle\chi \leftrightarrow \psi_1 \wedge \chi$

$(?)$ $\quad AX1 \overset{P}{\vdash} \langle\psi_1?\rangle \bar{x} \doteq \bar{x}' \leftrightarrow \psi_1 \wedge \bar{x} \doteq \bar{x}'$

$(T_1);(MP),(G2)$ $\quad AX1 \overset{P}{\vdash} \exists\bar{x}'(\chi[\bar{x}'/\bar{x}] \wedge \langle\psi_1?\rangle \bar{x} \doteq \bar{x}') \leftrightarrow$

$\exists\bar{x}'(\chi[\bar{x}'/\bar{x}] \wedge \psi_1 \wedge \bar{x} \doteq \bar{x}')$

$(T_1),(MP),(G2)$ $\quad AX1 \overset{P}{\vdash} \exists\bar{x}'(\chi[\bar{x}'/\bar{x}] \wedge \psi_1 \wedge \bar{x} \doteq \bar{x}') \leftrightarrow \chi \wedge \psi_1$

$(T_o),(MP)$ $\quad AX1 \overset{P}{\vdash} \exists\bar{x}'(\chi[\bar{x}'/\bar{x}] \wedge \langle\psi_1?\rangle \bar{x} \doteq \bar{x}') \leftrightarrow \chi \wedge \psi_1$

$(T_o),(MP)$ $\quad AX1 \overset{P}{\vdash} \langle\psi_1?\rangle\chi \leftrightarrow \exists\bar{x}'(\chi[\bar{x}'/\bar{x}] \wedge \langle\psi_1?\rangle \bar{x} \doteq \bar{x}')$

• $\alpha = (\beta \cup \beta')$

(a) $\varphi^d = (\langle\beta\rangle\psi \vee \langle\beta'\rangle\psi)^d = (\langle\beta\rangle\psi)^d \vee (\langle\beta'\rangle\psi)^d$

Con las hipótesis de inducción aplicadas a $\langle\beta\rangle\psi$ y a $\langle\beta'\rangle\psi$ y

$(T_0),(MP)$ $\quad AX1 \vdash^{P} (\langle\beta\rangle\psi)^d \vee (\langle\beta'\rangle\psi)^d \leftrightarrow \langle\beta\rangle\psi \vee \langle\beta'\rangle\psi$

$(\cup)$ $\quad AX1 \vdash^{P} \langle\beta\rangle\psi \vee \langle\beta'\rangle\psi \leftrightarrow \langle\beta\cup\beta'\rangle\psi$

$(T_0),(MP)$ $\quad AX1 \vdash^{P} \varphi \leftrightarrow \varphi^d$

(b)

$(\cup)$ $\quad AX1 \vdash^{P} \langle\beta\cup\beta'\rangle\chi \leftrightarrow \langle\beta\rangle\chi \vee \langle\beta'\rangle\chi$

Por hipótesis de inducción $\begin{cases} AX1 \vdash^{P} \langle\beta\rangle\chi \leftrightarrow \exists\bar{x}'(\chi[\bar{x}'/\bar{x}] \wedge \langle\beta\rangle\bar{x} \doteq \bar{x}') \\ AX1 \vdash^{P} \langle\beta'\rangle\chi \leftrightarrow \exists\bar{x}'(\chi[\bar{x}'/\bar{x}] \wedge \langle\beta'\rangle\bar{x} \doteq \bar{x}') \end{cases}$

$(T_1),(MP),(G2)$ $\quad AX1 \vdash^{P} \langle\beta\cup\beta'\rangle\chi \leftrightarrow \exists\bar{x}'(\chi[\bar{x}'/\bar{x}] \wedge (\langle\beta\rangle\bar{x} \doteq \bar{x}' \vee \langle\beta'\rangle\bar{x} \doteq \bar{x}'))$

$(\cup),(T_0),(MP)$ $\quad AX1 \vdash^{P} \langle\beta\cup\beta'\rangle\chi \leftrightarrow \exists\bar{x}'(\chi[\bar{x}'/\bar{x}] \wedge \langle\beta\cup\beta'\rangle\bar{x} \doteq \bar{x}')$

· $\alpha = (\beta;\beta')$

(a) $\quad \varphi^d = (\langle\beta\rangle\langle\beta'\rangle\psi)^d = (\langle\beta\rangle(\langle\beta'\rangle\psi)^d)^d$

Por hipótesis de inducción aplicada a $\langle\beta'\rangle\psi$

$AX1 \vdash^{P} \langle\beta'\rangle\psi \leftrightarrow (\langle\beta'\rangle\psi)^d$

$(T_0),(MP),(G1)$ $\quad AX1 \vdash^{P} \langle\beta\rangle\langle\beta'\rangle\psi \leftrightarrow \langle\beta\rangle(\langle\beta'\rangle\psi)^d$

Por hipótesis de inducción aplicada a $\langle\beta\rangle(\langle\beta'\rangle\psi)^d$

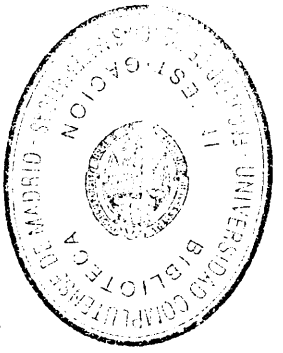
$AX1 \vdash^{P} \langle\beta\rangle(\langle\beta'\rangle\psi)^d \leftrightarrow (\langle\beta\rangle(\langle\beta'\rangle\psi)^d)^d$

$(T_0),(MP)$ $\quad AX1 \vdash^{P} \langle\beta\rangle\langle\beta'\rangle\psi \leftrightarrow (\langle\beta\rangle(\langle\beta'\rangle\psi)^d)^d$

$(T_0),(MP),(;)$ $\quad AX1 \vdash^{P} \varphi \leftrightarrow \varphi^d$

(b)

$(;)$ $\quad AX1 \vdash^{P} \langle\beta;\beta'\rangle\chi \leftrightarrow \langle\beta\rangle\langle\beta'\rangle\chi$



Por hipótesis de inducción para β'

$$AX1 \vdash^{P} \langle\beta'\rangle\chi \leftrightarrow \exists\bar{x}'(\chi[\bar{x}'/\bar{x}] \wedge \langle\beta'\rangle\bar{x} \doteq \bar{x}') \qquad (1)$$

Por hipótesis de inducción de (a) aplicada a $\langle\beta'\rangle\chi$

$$AX1 \vdash^{P} \langle\beta'\rangle\chi \leftrightarrow (\langle\beta'\rangle\chi)^{d}$$

$(T_o),(MP),(G1)$ $$AX1 \vdash^{P} \langle\beta\rangle\langle\beta'\rangle\chi \leftrightarrow \langle\beta\rangle(\langle\beta'\rangle\chi)^{d}$$

Por hipótesis de inducción para β

$$AX1 \vdash^{P} \langle\beta\rangle(\langle\beta'\rangle\chi)^{d} \leftrightarrow \exists\bar{x}''((\langle\beta'\rangle\chi)^{d}[\bar{x}''/\bar{x}] \wedge \langle\beta\rangle\bar{x} \doteq \bar{x}'')$$

Usando (1) y los axiomas de la lógica de primer orden $((T_1),(MP)$ y (G2)) tenemos $$AX1 \vdash^{P} \langle\beta;\beta'\rangle\chi \leftrightarrow \exists\bar{x}''(\exists\bar{x}'(\chi[\bar{x}'/\bar{x}] \wedge (\langle\beta'\rangle\bar{x} \doteq \bar{x}')^{d}[\bar{x}''/\bar{x}]) \wedge \langle\beta\rangle\bar{x} \doteq \bar{x}'')$$

$(T_1),(MP),(G2)$ $$AX1 \vdash^{P} \langle\beta;\beta'\rangle\chi \leftrightarrow \exists\bar{x}'(\chi[\bar{x}'/\bar{x}] \wedge \exists\bar{x}''((\langle\beta'\rangle\bar{x} \doteq \bar{x}')^{d}[\bar{x}''/\bar{x}] \wedge \langle\beta\rangle\bar{x} \doteq \bar{x}'')) \qquad (2)$$

Además

$$(;) \quad AX1 \vdash^{P} \langle\beta;\beta'\rangle\bar{x} \doteq \bar{x}' \leftrightarrow \langle\beta\rangle\langle\beta'\rangle\bar{x} \doteq \bar{x}'$$

Por hipótesis de inducción de (a) aplicada a $\langle\beta'\rangle\bar{x} \doteq \bar{x}'$

$$AX1 \vdash^{P} \langle\beta'\rangle\bar{x} \doteq \bar{x}' \leftrightarrow (\langle\beta'\rangle\bar{x} \doteq \bar{x}')^{d}$$

Y por hipótesis de inducción aplicada a β y con $(T_o),(MP)$ tenemos $$AX1 \vdash^{P} \langle\beta;\beta'\rangle\bar{x} \doteq \bar{x}' \leftrightarrow \exists\bar{x}''((\langle\beta'\rangle\bar{x} \doteq \bar{x}')^{d}[\bar{x}''/\bar{x}] \wedge \langle\beta\rangle\bar{x} \doteq \bar{x}'')$$

Con esto y (2) obtenemos con $(T_1),(MP),(G2)$

$$AX1 \vdash^{P} \langle\beta;\beta'\rangle \leftrightarrow \exists\bar{x}'(\chi[\bar{x}'/\bar{x}] \wedge \langle\beta;\beta'\rangle\bar{x} \doteq \bar{x}')$$

· $\alpha = \beta^*$

(a) $\varphi^d = \exists z\, \exists \bar{y}(\underline{traza}^d_{\beta^*}(z,\bar{y}) \wedge \bar{y}(0') \doteq \bar{x} \wedge \psi^d[\bar{y}(z)/\bar{x}])$

con $\underline{traza}^d_{\beta^*}(z,\bar{y}) = \forall z_1\ (z_1 \leq' z \longrightarrow \exists \bar{x}(\bar{x} \doteq \bar{y}(z_1) \wedge (\langle\beta\rangle \bar{x} \doteq \bar{y}(suc'(z_1)))^d$

Hemos de probar

(i) $AX1 \vdash^P \varphi \longrightarrow \varphi^d$

(ii) $AX1 \vdash^P \varphi^d \longrightarrow \varphi$

y con (T_o),(MP) concluimos el resultado.

Prueba de (i)

$$AX1 \vdash^P \langle\beta^*\rangle\psi \longrightarrow \varphi^d$$

Lo haremos usando una regla dual a (I). Es fácil probar que la regla

(I') $\dfrac{\langle\alpha\rangle\eta \longrightarrow \eta}{\langle\alpha^*\rangle\eta \longrightarrow \eta}$ es válida en el cálculo P (cfr. Lema 1.6.1)

Para aplicarla probaremos

$$AX1 \vdash^P \langle\beta\rangle\varphi^d \longrightarrow \varphi^d$$

Por hipótesis de inducción de (b) tenemos

$$AX1 \vdash^P \langle\beta\rangle\varphi^d \longleftrightarrow \exists \bar{x}'(\varphi^d[\bar{x}'/\bar{x}] \wedge \langle\beta\rangle \bar{x} \doteq \bar{x}')$$

Además por las propiedades de la función beta de Gödel, deducibles de AX1 $AX1 \vdash^P \exists \bar{y}'(\bar{y}'(0') \doteq \bar{x} \wedge \forall z_1(z_1 \leq' z \longrightarrow \bar{y}'(suc'(z_1)) \doteq \bar{y}(z_1)))$

es decir, podemos ampliar la "traza" $\bar{y}$ concatenando $\bar{x}$ como primer elemento.

Claramente empleando dicha $\bar{y}'$ se tiene

$(T_1),(MP),(G2)\quad AX1 \vdash^{P} \langle\beta\rangle\varphi^{d} \longrightarrow \exists \bar{y}'(\underline{traza}^{d}_{\beta *}(suc'(z),\bar{y}') \wedge$

$\bar{y}'(0') \doteq \bar{x} \wedge \psi^{d}[\bar{y}'(suc'(z))/\bar{x}])$

y por tanto

$(T_1),(MP),(G2)\quad AX1 \vdash^{P} \langle\beta\rangle\varphi^{d} \longrightarrow \varphi^{d}$

Por la regla (I') tenemos

$$AX1 \vdash^{P} \langle\beta^{*}\rangle\varphi^{d} \longrightarrow \varphi^{d} \qquad (1)$$

Además, también por las propiedades de la función beta, tenemos

$(T_1),(MP),(G2)\quad AX1 \vdash^{P} \exists\bar{y}(\bar{y}(0') \doteq \bar{x})$ y

$(T_1),(MP),(G2)\quad AX1 \vdash^{P} \psi^{d} \longrightarrow \exists\bar{y}(\underline{traza}^{d}_{\beta *}(0',\bar{y}) \wedge \bar{y}(0') \doteq \bar{x}$

$\wedge \psi^{d}[\bar{y}(0')/\bar{x}])$

Por hipótesis de inducción

$$AX1 \vdash^{P} \psi \longleftrightarrow \psi^{d}$$

y obtenemos

$(T_0),(MP)\quad AX1 \vdash^{P} \psi \longrightarrow \varphi^{d}$

Con (G1) $\quad AX1 \vdash^{P} \langle\beta^{*}\rangle\psi \longrightarrow \langle\beta^{*}\rangle\varphi^{d}$

y con (1),(MP),(T_0) concluimos

$$AX1 \vdash^{P} \langle\beta^{*}\rangle\psi \longrightarrow \varphi^{d}$$

Prueba de (ii)

$$AX1 \vdash^{P} \varphi^{d} \longrightarrow \langle\beta^{*}\rangle\psi$$

Consideremos la fórmula

$\varphi_1 = \exists\bar{y}(\underline{traza}^{d}_{\beta *}(z,\bar{y}) \wedge \bar{y}(0') \doteq \bar{x} \wedge \psi^{d}[\bar{y}(z)/\bar{x}])$

que tiene a z entre sus variables libres

Usaremos la regla (C). Para ello probaremos

$$AX1 \vdash^{P} \varphi_1[suc'(z)/z] \longrightarrow <\beta> \varphi_1$$

$$\varphi_1[suc'(z)/z] = \exists \bar{y}(\underline{traza}^{d}_{\beta *}(suc'(z),\bar{y}) \wedge \bar{y}(0') \doteq \bar{x} \wedge \psi^{d}[\bar{y}(suc'(z))/\bar{x}])$$

Por las propiedades de la función beta tenemos

$$(T_1),(MP),(G2) \quad AX1 \vdash^{P} \exists \bar{y}' \forall z_1 (z_1 \leq' z \longrightarrow \bar{y}'(z_1) \doteq \bar{y}(suc'(z_1)))$$

Es decir, podemos simplificar la traza quitandole el primer elemento. Claramente empleando dicha $\bar{y}'$ resulta

$$(T_1),(MP),(G2) \quad AX1 \vdash^{P} \varphi_1[suc(z)/z] \longrightarrow \exists \bar{y}'(\underline{traza}^{d}_{\beta *}(z,\bar{y}') \wedge \bar{y}'(0') \doteq \bar{y}(suc'(0')) \wedge \psi^{d}[\bar{y}'(z)/\bar{x}]) \wedge <\beta>\bar{x} \doteq \bar{y}(suc'(0'))$$

y tomando $\bar{x}' = \bar{y}(suc'(0'))$ tenemos

$$(T_1),(MP),(G2) \quad AX1 \vdash^{P} \varphi_1[suc'(z)/z] \longrightarrow \exists \bar{x}'(\varphi_1[\bar{x}'/\bar{x}] \wedge <\beta>\bar{x} \doteq \bar{x}')$$

y por hipótesis de inducción de (b) y $(T_o),(MP)$

$$AX1 \vdash^{P} \varphi_1[suc'(z)/z] \longrightarrow <\beta> \varphi_1$$

Con la regla (C) tenemos

$$AX1 \vdash^{P} \varphi_1 \longrightarrow <\beta^{*}> \varphi_1[0'/z]$$

Además

$$(T_o) \quad AX1 \vdash^{P} \varphi^{d} \longrightarrow \exists z \varphi_1 \qquad y$$

$$(T_o),(MP),(G2) \quad AX1 \vdash^{P} \varphi^{d} \longrightarrow <\beta^{*}> \varphi_1[0'/z]$$

Como

$$(T_1),(MP),(G2) \quad AX1 \vdash^{P} \varphi_1[0'/z] \longrightarrow \psi^{d}$$

y por hipótesis de inducción

$$AX1 \vdash^{P} \psi \longleftrightarrow \psi^{d}$$

tenemos

$(T_0),(MP),(G1)$ $\quad AX1 \vdash^{P} \langle \beta^{*} \rangle \varphi_1[0'/z] \longrightarrow \langle \beta^{*} \rangle \psi$

y por tanto

$(T_0),(MP)$ $\quad AX1 \vdash^{P} \varphi^{d} \longrightarrow \langle \beta^{*} \rangle \psi$

(b)

Si aplicamos la hipótesis de inducción de (a) a $\langle \beta^{*} \rangle \chi$ y $\langle \beta^{*} \rangle \bar{x} \doteq \bar{x}'$ tenemos

$$AX1 \vdash^{P} \langle \beta^{*} \rangle \chi \longleftrightarrow \exists z \exists \bar{y}(\underline{traza}^{d}_{\beta^{*}}(z,\bar{y}) \wedge \bar{y}(0') \doteq \bar{x} \wedge \chi[\bar{y}(z)/\bar{x}])$$

$$AX1 \vdash^{P} \langle \beta^{*} \rangle \bar{x} \doteq \bar{x}' \longleftrightarrow \exists z \exists \bar{y}(\underline{traza}^{d}_{\beta^{*}}(z,\bar{y}) \wedge \bar{y}(0') \doteq \bar{x} \wedge \bar{y}(z) \doteq \bar{x}')$$

Ahora tomando $\bar{x}' = \bar{y}(z)$ concluimos

$(T_1),(MP),(G2)$ $\quad AX1 \vdash^{P} \langle \beta^{*} \rangle \chi \longleftrightarrow \exists \bar{x}'(\chi[\bar{x}'/\bar{x}] \wedge \langle \beta^{*} \rangle \bar{x} \doteq \bar{x}')$

Con estos tres lemas estamos en condiciones de probar lo que queríamos.

Supongamos $AX2 \cup \Phi \vdash^{NP} \varphi$. Existirá entonces una sentencia θ , conjunción finita de cierres universales de ciertas fórmulas de Φ, tal que $AX2 \vdash^{NP} \theta \longrightarrow \varphi$. Por la corrección del cálculo NP tenemos $AX2 \models \theta \longrightarrow \varphi$.

Ahora $AX1 \models (\theta \longrightarrow \varphi)^{d}$, ya que si $\mathfrak{D} \models AX1$ entonces $\mathfrak{M}_{\mathfrak{D}} \models AX2$ por el lema 2,4.2, y por tanto $\mathfrak{M}_{\mathfrak{D}} \models (\theta \longrightarrow \varphi)$ y $\mathfrak{D} \models (\theta \longrightarrow \varphi)^{d}$ por el lema 2.4.1

Como el cálculo P encierra un cálculo completo para la lógica de primer orden y $(\Theta \longrightarrow \varphi)^d \in F_d$ obtenemos

$$AX1 \vdash^{P} (\Theta \longrightarrow \varphi)^d$$

Con el lema 2.4.3 y la regla (MP)

$$AX1 \vdash^{P} \Theta \longrightarrow \varphi$$

y concluimos $AX1 \cup \Phi \vdash^{P} \varphi$ por la elección de Θ

Este teorema caracteriza el cálculo P en NDL, o más exactamente la derivabilidad en P a partir de $PA_d \cup DIA_d$. Es posible interpretar el teorema como un resultado de completitud, de la siguiente forma: Si llamamos admisibles a las estructuras $\mathcal{M}$ de tipo td que sean modelo de $TIA_{td} \cup PA_t \cup DIA_{td} \cup EXT^b$, el teorema garantiza para $\Phi \subseteq SDF_d$, $\varphi \in SDF_d$ cualesquiera, que $PA_d \cup DIA_d \cup \Phi \vdash^{P} \varphi$ si y solo si φ es válida en todo modelo admisible $\mathcal{M}$ de $PA_d \cup DIA_d \cup \Phi$.

Otros resultados en esta línea pueden encontrarse en (2),(33). Caracterizan dentro de NDL el método modal de Burstall, la lógica temporal de Pnueli, el método de Floyd - Hoare y el método de Manna-Cooper.

Un sistema razonable de axiomas para el trabajo con NDL puede ser el siguiente:

Definición 2.4.5.- DAX

Sea d tipo de similaridad que contiene los símbolos 0' y suc'

$$DAX = PA_t \cup TIA_{td} \cup DIA_{td} \cup EXT$$

En (34) puede encontrarse el siguiente resultado para DAX

$$\vdash^{P} \quad \underset{SDF}{<} \quad (DAX \vdash^{NP})$$

Más aún, existen una fórmula φ y un programa α tales que

$$\not\vdash^{P} [\alpha]\varphi \qquad y \qquad DAX \vdash^{NP} [\alpha]\varphi$$

lo cual demuestra que el cálculo NP usando a DAX como axiomas es más potente que P. De hecho la potencia deductiva de P para fórmulas de la forma $\varphi \longrightarrow [\alpha]\psi$ es la del cálculo de Hoare, y el resultado recien citado implica en particular la conocida incompletitud de éste (cfr. (18)).

2.5.- Una lógica de la programación de primer orden para programas recursivos.

Cartwright en (6) propuso el uso de la lógica de primer orden como base para razonar acerca de programas recursivos. Esta sección supone una concisa exposición de sus planteamientos, ligeramente modificados para adoptarlos a nuestro contexto y los cuales intentaremos relacionar con las ideas no standar de NDL, en el siguiente capítulo.

Definición 2.5.1.-

(1) Un orden parcial completo sobre un conjunto S es una relación binaria $\sqsubseteq$ sobre S con

(i) $\sqsubseteq$ es un orden parcial sobre S

(ii) S contiene un elemento mínimo para la relación $\sqsubseteq$

(iii) Cualquier subconjunto X de S que sea dirigido (en el sentido de que para todo $x,y \in X$ existe $z \in X$ con $x \sqsubseteq z, y \sqsubseteq z$) posee una cota superior mínima $\bigsqcup X$ en S

$(S, \sqsubseteq)$ se denomina c.p.o. (orden parcial completo).

(2) Sean $(S_1, \sqsubseteq_1)$, $(S_2, \sqsubseteq_2)$ c.p.o.s.

$f : S_1 \longrightarrow S_2$ es continua si cumple que para todo $X \subseteq S_1$ dirigido $f(X)$ es dirigido y $f(\bigsqcup_1 X) = \bigsqcup_2 f(X)$

Un resultado conocido para c.p.o. y funciones continuas es el teorema de Kléene:

Teorema de Kleene: Si $(S, \sqsubseteq)$ es un c.p.o. con elemento mínimo $\perp$ y $f : S \longrightarrow S$ es continua, entonces f tiene un mínimo punto fijo $f\uparrow\omega$ calculable como $\bigsqcup_{n\in N} f^n(\perp)$.

Este teorema puede encontrarse en (29) y se remonta a ideas de Tarski (cfr. (43)), quien demostró una proposición más general para operadores monótonos.

Siguiendo a Cartwright, atribuiremos a Kleene la forma particularmente simple que toma el resultado cuando el operador es continuo.

Definición 2.5.2.-

Sea $\mathcal{D}$ una estructura de tipo de similaridad d.

Un programa recursivo sobre $\mathcal{D}$ es un conjunto finito de ecuaciones $F = \{ f_1(\bar{x}_1) = \tau_1, \ldots, f_n(\bar{x}_n) = \tau_n \}$ con $n > 0$, $f_1, \ldots, f_n$ símbolos de función que no aparecen en d y cada f_i de m_i argumentos, y $\tau_1, \ldots, \tau_n \in TR_d$, tal que τ_i solo utiliza las variables $\bar{x}_i$

TR_d se define recursivamente como:

- $x_i^j \in TR_d$
- $g(\sigma_1, \ldots, \sigma_m) \in TR_d$ si $\sigma_1, \ldots, \sigma_m \in TR_d$ y g es una función de m argumentos de d
- $f_i(\sigma_1, \ldots, \sigma_{m_i}) \in TR_d$ si $\sigma_1, \ldots, \sigma_{m_i} \in TR_d$, $1 \leq i \leq n$
- if $Q\, \sigma_1, \ldots, \sigma_m$ then τ'_1 else $\tau'_2 \in TR_d$ si $\sigma_1, \ldots, \sigma_m, \tau'_1, \tau'_2 \in TR_d$

Q símbolo de predicado de m argumentos de d

Este último término corresponde al <u>if</u>--<u>then</u>--<u>else</u>-- habitual.

Si $\tau = \text{if } Q\ \sigma_1,\ldots,\sigma_m \text{ then } \tau'_1 \text{ else } \tau'_2$ tendremos $\tau^{\mathfrak{D}} = \tau'^{\mathfrak{D}}_1$ si se verifica $Q^{\mathfrak{D}}\sigma_1,\ldots,\sigma_m$ y $\tau^{\mathfrak{D}} = \tau'^{\mathfrak{D}}_2$ si se verifica $\neg\, Q^{\mathfrak{D}}\sigma_1,\ldots,\sigma_m$

Cada programa recursivo F sobre $\mathfrak{D}$ lleva asociado un operador $T_F^{\mathfrak{D}}$

$$T_F^{\mathfrak{D}} : \mathcal{P}(D^{m_1+1}) \times \ldots \times \mathcal{P}(D^{m_n+1}) \longrightarrow \mathcal{P}(D^{m_1+1}) \times \ldots \times \mathcal{P}(D^{m_n+1})$$

con $T_F^{\mathfrak{D}}(A_1,\ldots,A_n) = (A'_1,\ldots,A'_n)$ si

$A'_i = \{ (a_1,\ldots,a_{m_i},b) \;/\; a_1,\ldots,a_{m_i} \in D$ y existe $b \in D$ que se obtiene de calcular τ_i en $\mathfrak{D}$ sustituyendo $\bar{x}_i$ por $a_1,\ldots,a_{m_i}$ y cada aparición de $f_j(\bar{c})$ por $c' \in D$ con $(c_1,\ldots,c_{m_j},c') \in A_j \}$

Si $T_F^{\mathfrak{D}}$ opera sobre funciones parciales entendidas como grafos, el resultado es a su vez el grafo de una serie de funciones parciales.

Si consideramos $\mathcal{P}(D^{m_1+1}) \times \ldots \times \mathcal{P}(D^{m_n+1})$ y la relación binaria definida por

$$A_1,\ldots,A_n \sqsubseteq B_1,\ldots,B_n \iff A_1 \subseteq B_1,\ldots,A_n \subseteq B_n$$

$(\mathcal{P}(D^{m_1+1}) \times \ldots \times \mathcal{P}(D^{m_n+1}),\ )$ es un c.p.o. (el elemento mínimo es $(\emptyset,\ldots,\emptyset)$ y la cota superior de conjuntos dirigidos se obtiene con la $\bigcup$ conjuntista), y $T_F^{\mathfrak{D}}$ es continuo.

Por el teorema de Kleene podemos considerar $\mathbf{F} = \{\mathbf{F}_1,\ldots,\mathbf{F}_n\}$ las relaciones de $T_F^{\mathfrak{D}} \uparrow \omega$ mínimo punto fijo de $T_F^{\mathfrak{D}}$

$\mathbf{F}$ puede considerarse la semántica del programa recursivo F, tomando cada $\mathbf{F}_i$ como el grafo de la función f_i de F. Podemos conside-

rar el tipo de similaridad $d^+ = d \cup \{F_1,\ldots,F_n\}$ con $F_1,\ldots,F_n$ símbolos de predicado que no aparecen en d con F_i de m_i+1 argumentos, y $\mathfrak{D}^+$ la estructura de tipo de similaridad d^+ con igual dominio que $\mathfrak{D}$ y que interpreta los símbolos de d como en $\mathfrak{D}$ y $F_1,\ldots,F_n$ como **F**.

En $\mathfrak{D}^+$ podemos probar ciertas propiedades del programa F. Si $\mathfrak{D}$ cumple unos ciertos axiomas $AX_{\mathfrak{D}}$ podemos utilizar $AX_{\mathfrak{D}}$ y las ecuaciones de F para probar dentro de la lógica de primer orden propiedades de **F**.

Esto es lo que Cartwright llama "lógica de la programación de primer orden".

En particular Cartwright enuncia un resultado para estructuras $\mathfrak{D}$ accesibles(todo elemento de D es el valor en $\mathfrak{D}$ de un término cerrado de tipo d. Estas estructuras se estudiaran con más detalle en el capítulo 3).

En estas estructuras se verifica un axioma de inducción $\underline{ind}_d(\varphi)$ para relaciones $\varphi(x)$ sobre $\mathfrak{D}$

$$\underline{ind}_d(\varphi) = \Big(\bigwedge_{g \in d} \forall x_1,\ldots,x_{\#g}\big(\varphi[x_1/x] \wedge \ldots \wedge \varphi[x_{\#g}/x] \longrightarrow \varphi[g(x_1,\ldots,x_{\#g})/x]\big) \longrightarrow \forall x\,\varphi$$

(donde # g indica el número de argumentos de la función g).

El teorema fundamental de Cartwright afirma que en $\mathfrak{D}^+$ se verifica el axioma de inducción $\underline{ind}_d(\varphi)$ para cualquier fórmula φ de tipo d^+.

La prueba es inmediata dado que el dominio de $\mathfrak{D}^+$ es el mismo de $\mathfrak{D}$, luego todo elemento de $\mathfrak{D}^+$ es el valor de un término cerrado, y por ello el axioma de inducción se satisface para cualquier propiedad φ, en particular para las definibles mediante fórmulas de tipo d^+.

La importancia de este resultado radica en que el principio de inducción puede usarse en $\mathfrak{D}^+$ para establecer en $\mathfrak{D}^+$ propiedades de las funciones introducidas por el programa F.

Un ejemplo de ésto es el siguiente:

Consideremos la estructura de las listas generales

$\mathfrak{D} = (D, 1^{\mathfrak{D}}, \phi^{\mathfrak{D}}, \underline{car}^{\mathfrak{D}}, \underline{cdr}^{\mathfrak{D}}, \underline{lista}^{\mathfrak{D}}, \underline{átomo}^{\mathfrak{D}})$ donde $1^{\mathfrak{D}}$ es la función de dos argumentos que genera listas a partir de átomos o listas, $\underline{lista}^{\mathfrak{D}}$ y $\underline{átomo}^{\mathfrak{D}}$ son predicados de un argumento que deciden si un miembro de D es lista o es átomo. $\underline{lista}^{\mathfrak{D}}$ y $\underline{atomo}^{\mathfrak{D}}$ dividen D en dos partes disjuntas. $\phi^{\mathfrak{D}}$ es una constante que indica la lista vacia y $\underline{car}^{\mathfrak{D}}$ y $\underline{cdr}^{\mathfrak{D}}$ son las proyecciones de $1^{\mathfrak{D}}$. Abreviaremos 1(x,y) como x.y

En concreto en $\mathfrak{D}$ se verifican los siguientes axiomas:

A0: $\forall x\ (\underline{átomo}(x) \vee \underline{lista}(x))$

A1: $\underline{lista}(\phi)$

A2: $\forall x \forall y\ (\underline{lista}(x.y) \leftrightarrow \underline{lista}(y))$

A3: $\forall x \forall y\ \underline{car}(x.y) \doteq x$

A4: $\forall x \forall y\ \underline{cdr}(x.y) \doteq y$

A5: $(\varphi[\phi/x] \wedge \forall x(\underline{\text{átomo}}(x) \longrightarrow \varphi) \wedge \forall x \forall x'(\varphi \wedge \varphi[x'/x] \longrightarrow \varphi[x.x'/x])) \longrightarrow \forall x \varphi$

Consideremos el programa recursivo

$F = \{ rel(x) = rel1(x, \phi)$

$rel1(x,z) = \underline{\text{if}}\ x \doteq \phi\ \underline{\text{then}}\ z$

$\underline{\text{else}}\ (\underline{\text{if}}\ \underline{\text{átomo}}(x)\ \underline{\text{then}}\ x.z$

$\underline{\text{else}}$

$rel1(car(x),rel1(cdr(x),z)))$

que genera una lista rel(x) con todos los átomos de la lista x

Si $x = (a.((b.c).d).\phi))$ tenemos $rel(x) = (a.(b.(c.d)))$

Entre las propiedades que podemos probar acerca de F está la terminación del programa.

Probemos que $\forall x \exists y\ (\underline{\text{lista}}(y) \wedge Rel(x,y))$

Para ello basta probar que

$$\forall x \forall z\ (\underline{\text{lista}}(z) \longrightarrow \exists y(\underline{\text{lista}}(y) \wedge Rel1(x,z,y)))$$

y lo haremos usando el axioma de inducción A5 para

$\varphi = \forall z(\underline{\text{lista}}(z) \longrightarrow \exists y(\underline{\text{lista}}(y) \wedge Rel1(x,z,y)))$

$\varphi[\phi/x]$

$rel1(\phi,z) = z$, luego tomando $z = y$ tenemos

$$\forall z(\underline{\text{lista}}(z) \longrightarrow \exists y(\underline{\text{lista}}(y) \wedge Rel1(\phi,z,y)))$$

$\forall x(\underline{\text{átomo}}(x) \longrightarrow \varphi)$

Si $\underline{\text{átomo}}(x)$ supone $rel1(x,z) = x.z$, luego tomando $y = x.z$ tenemos

$\underline{\text{átomo}}(x) \longrightarrow \forall z(\underline{\text{lista}}(z) \longrightarrow \exists y(\underline{\text{lista}}(y) \wedge Rel1(x,z,y)))$

$\forall x\, \forall x'(\varphi \wedge \varphi[x'/x] \longrightarrow \varphi[x.x'/x])$

Por A2 $\underline{\text{lista}}(x.x') \longrightarrow \underline{\text{lista}}(x')$

Por A4 $\underline{\text{cdr}}(x.x') = x'$

y con la hipótesis de inducción $\varphi[x'/x]$ tenemos

$\forall x\, \forall x'(\varphi \wedge \varphi[x'/x] \longrightarrow \forall z(\underline{\text{lista}}(z) \longrightarrow \exists y'(\underline{\text{lista}}(y') \wedge \text{Rel1}(\underline{\text{cdr}}(x,x'),z,y')))$

Por A3 $\underline{\text{car}}(x.x') = x$

y con la hipótesis de inducción φ tenemos

$\forall x\, \forall x'(\varphi \wedge \varphi[x'/x] \longrightarrow \forall z(\underline{\text{lista}}(z) \longrightarrow \exists y\, \exists y'(\underline{\text{lista}}(y) \wedge \underline{\text{lista}}(y') \wedge \text{Rel1}(\underline{\text{car}}(x.x'),y',y) \wedge \text{Rel1}(\underline{\text{cdr}}(x.x'),z,y')))$

Y como $\text{rel1}(x.x',z) = \text{rel1}(\underline{\text{car}}(x.x'),\text{rel1}(\underline{\text{cdr}}(x.x'),z))$ tenemos

$\forall x\, \forall x'(\varphi \wedge \varphi[x'/x] \longrightarrow \varphi[x.x'/x])$

Concluimos $\forall x\, \forall z(\underline{\text{lista}}(z) \longrightarrow \exists y(\underline{\text{lista}}(y) \wedge \text{Rel1}(x,z,y)))$

En todo caso considerar como semántica de programas recursivos el mínimo punto fijo del operador $T_F^{\mathfrak{D}}$ tiene sus problemas.

Consideremos el programa recursivo

$F = \{\text{cero}(x) = \underline{\text{if}}\ x \doteq 0\ \underline{\text{then}}\ 0\ \underline{\text{else}}\ \text{cero}(x-1)\}$

para la estructura accesible $\mathfrak{N}$ modelo standard de la aritmética.

Sea $\mathfrak{N}'$ un modelo no standard de $\text{Th}(\mathfrak{N})$.

Con el axioma de inducción $\underline{\text{ind}}_d$ para $\mathfrak{N}$ podemos probar con la lógica de primer orden que $\forall x\ \text{cero}(x) \doteq 0$ (o más exactamente que $\forall x\ \text{Cero}(x,0)$).

Sin embargo el mínimo punto fijo de $T_F^{\mathfrak{D}}$ nos da la función par-

cial (su grafo) **Cero**(x,0) solo para los x standard.Como el conjunto de estos no es definible en $\mathcal{N}$ los axiomas de inducción que justifican la demostración de $\forall x$ Cero(x,0) no valen en $\mathcal{N}$

Esto parece apoyar las conocidas críticas de Hitchock y Park (cfr. (21)), en el sentido de que "terminación" y "totalidad" no son nociones de primer orden.

Para conciliar esto con la intuición de que, pese a todo, $\forall x$ Cero(x,0) es demostrable por inducción sobre x, puede alterarse la interpretación de los programas recursivos.

Para ello Cartwright introduce la noción de mínimo punto fijo definible. La interpretación alternativa de las funciones del programa seran las funciones parciales determinadas por el mínimo punto fijo definible de T_F .

Asi se obtiene un resultado que generaliza el teorema fundamental. Como, además, el mínimo punto fijo de T_F es definible en estructuras standard el teorema fundamental se mantendrá como un caso particular de su generalización.

En todo caso, el mínimo punto fijo definible del operador $T_F^{\mathfrak{D}}$ no siempre existe.

La solución a este problema ofrecida por Cartwright es trabajar con una clase de estructuras, las estructuras que soportan sintaxis

que en esencia aseguran la existencia de mecanismos de codificación que representan en D secuencias finitas de elementos de D (y este concepto de finitud es "en $\mathfrak{D}$", y puede no corresponder con la noción intuitiva desde "fuera de $\mathfrak{D}$"). Estas estructuras son similares a las estructuras aritméticas de Harel (ver definición 1.4.1) aunque más generales.

Nosotros vamos a desarrollar esto de una manera más simple, heredando de NDL su noción de sucesiones en estructuras de tres generos. Trabajaremos con estructuras de tipo td

$$\mathfrak{M} = (\mathcal{T} , \mathfrak{D} , S, ext^{\mathfrak{M}})$$

con análogas definiciones a las de 2.2 pero restringiendo el tipo de similaridad t a los símbolos de la aritmética de Peano 0, 1, +. Veremos estas estructuras con más detalle en el capítulo 3.

Definición 2.5.3.-

$$AX_t = \{ A1 = (\forall i \neg (i+1 \doteq 0)),$$

$$A2 = (\forall i \forall j (i+1 \doteq j+1 \longrightarrow i \doteq j)),$$

$$A3 = (\forall i (i+0 \doteq i)),$$

$$A4 = (\forall i \forall j (i+(j+1) \doteq (i+j)+1))$$

AX_t son los axiomas a exigir a las estructuras temporales

$I_t = \{ \underline{ind}_t(\varphi) \ / \ \varphi \in F_{td} \}$ donde

$$\underline{ind}_t(\varphi) = (\varphi[0/i] \wedge \forall j (\varphi[j/i] \longrightarrow \varphi[j+1/i])) \longrightarrow \forall i \varphi$$

I_t son los axiomas de inducción para el tiempo para fórmulas de

primer orden de tipo td.

$AX_s = \{ \underline{unit}, \underline{conc} \}$ donde

$\underline{unit} = \forall x \exists u \forall i(u(i) \doteq x)$

$\underline{conc} \doteq \forall u \forall i \forall v \forall j \exists w \forall k((k \leq i \longrightarrow w(k) \doteq u(k)) \wedge$

$(i < k \leq i+j+1 \longrightarrow \exists l(l \leq j \wedge l+i+1 \doteq k \wedge w(k) \doteq v(l))))$

<u>unit</u> indica que existen todas las sucesiones constantes y <u>conc</u> que las sucesiones se pueden concatenar.

Son los axiomas que vamos a exigir para S.

Las relaciones $<$, y $\leq$ se definen:

$i < j \iff \exists k(i+(k+1) \doteq j)$

$i \leq j \iff (i < j \vee i \doteq j)$

<u>Definición 2.5.4.-</u> Estructuras admisibles

Una estructura admisible de tipo td $\mathfrak{M} = (\mathcal{T}, \mathcal{D}, S, ext^{\mathfrak{M}})$ es cualquier $\mathfrak{M}$, con $\mathfrak{M} \models AX_t \cup I_t \cup AX_s$

<u>Lema 2.5.1.-</u> (<u>Teorema de Kleene generalizado</u>)

Sea $\mathfrak{M}$ estructura admisible de tipo td y F un programa recursivo sobre $\mathcal{D}$

Entonces $T_F^{\mathcal{D}}$ tiene un mínimo punto fijo definible por una fórmula de F_{td}.

Una demostración a un resultado semejante se presenta en el capítulo 3, y por eso no incluimos esta . La prueba con la terminología de Cartwright puede encontrarse en (5), siendo más engorrosa que la

que nosotros daremos.

Teorema 2.5.1.-

Sea $\mathcal{M}$ una estructura admisible $\mathcal{M} = (\tau , \mathfrak{D} , S, ext^{\mathcal{M}})$ con $\mathcal{M} \models \{ \underline{ind}_d(\varphi) / \varphi \in F_{td} \}$.

Sea F un programa recursivo sobre $\mathfrak{D}$ y sea $\mathbf{F}'$ el mínimo punto fijo definible de T_F . Consideremos $\mathfrak{D}^+ = (\tau , \mathfrak{D}^+, S, ext^{\mathcal{M}})$ con $\mathfrak{D}^+$ construido como hemos visto (interpretando $F_1,\ldots,F_n$ como $\mathbf{F}'$). Entonces en $\mathcal{M}^+$ se verifica $\underline{ind}_d(\varphi)$ para fórmulas φ de tipo td^+.

Demostración.-

El lema 2.5.1 nos asegura la existencia de $\mathbf{F}'$. Como $\mathbf{F}'$ es definible por una fórmula de F_{td} se puede reemplazar en cualquier fórmula φ las apariciones de los símbolos $F_1,\ldots,F_n$ por las definiciones de $\mathbf{F}'_1,\ldots,\mathbf{F}'_n$, obteniendo una fórmula de F_{td} equivalente a φ en $\mathcal{M}^+$ para la que si se verifica el axioma de inducción por hipótesis.

Este teorema nos asegura la validez del axioma de inducción en estructuras admisibles para fórmulas que utilicen los símbolos $F_1,\ldots,F_n$. Así podemos utilizar dicho axioma para probar propiedades sobre el programa F. Para dichas pruebas usaremos axiomas para $\mathfrak{D}$, más los axiomas de inducción $\underline{ind}_d(\varphi)$ y las ecuaciones del programa F. Como se vio en el ejemplo los axiomas de inducción son decisivos en estas demostraciones, en especial al tratar de probar propiedades de totalidad.

Así pues la lógica de programas de primer orden es un sistema formal apropiada (y cómodo por la utilización de los axiomas de inducción) para razonar acerca de las propiedades de los programas recursivos.

En el capítulo siguiente se tomarán algunas de estas ideas para desarrollar una lógica para programación lógica. Se trabajará con predicados en lugar de con funciones y el marco de trabajo serán estructuras de tres géneros similares a las de 2.2. Con ello conseguimos una codificación de sucesiones de datos como "trazas internas" en la estructura (el conjunto de sucesiones S). Aparte de la simplificación formal, obtenemos algunas ventajas. Por ejemplo si consideramos el lenguaje de la aritmética y los programas recursivos

$F_1 = \{ g(x) \doteq g(x) \}$ y $F_2 = \{ g(x) \doteq g(x) + 1 \}$

ambos definen la función totalmente indefinida, por lo que se tiene

$\varphi = \forall x \forall y \neg G(x,y)$

Sin embargo de $AX_{\mathcal{N}}$ más los axiomas de inducción para fórmulas que usen G, más la ecuación de F_1 no puede inferirse φ, debido a que $T_{F_1}^{\mathcal{N}}$ admite otros puntos fijos. φ si puede inferirse para F_2 pues $T_{F_2}^{\mathcal{N}}$ tiene un único punto fijo.

Como veremos más adelante, la lógica propuesta en el capítulo 3 no tiene este problema.

Además Cartwright para resolverlo recurre a construir un programa recursivo F^*, con $T_{F^*}^{\mathcal{D}}$ con un único punto fijo definible el cual

determina el mínimo punto fijo definible de $T_F^{\mathcal{D}}$. Este programa F^* solo existe con esta propiedad en estructuras con representación interna de sucesiones de datos. Es decir, en estructuras admisibles en nuestra terminología. Dado que las "trazas internas" acaban siendo necesarias, es natural incluirlas en las estructuras.

Por otro lado las nociones de mínimo punto fijo y mínimo punto fijo definible volverán a utilizarse.

Todo esto es ilustrativo de las diferencias que hay entre usar semántica tradiccional o semántica no standard. Resumiendo la discusion anterior, consideramos que esta última es la que se corresponde adecuadamente con el empleo de formalismos de primer orden para derivar propiedades de programas.

CAPITULO 3: UNA LOGICA NO STANDARD PARA PROGRAMAS LOGICOS.

3.1.- Introducción: La lógica NPL para programación lógica.

Desde hace algún tiempo esta tomando importancia la programación lógica. Surge de los intentos para mecanizar el razonamiento lógico (confrontar los trabajos de Green (16)) y del desarrollo de la inteligencia artificial. Kowalski y Colmerauer ((28) y (8)) dieron la idea fundamental de que la lógica puede ser usada como lenguaje de programación. Se ha desarrollado, incluso, un lenguaje de programación concreto: el PROLOG (cuyo primer interprete fue desarrollado por Roussell (37)).

La programación lógica trabaja con los programas de Horn y dentro de interpretaciones de Herbrand.

En este capítulo proponemos una lógica para el razonamiento acerca de propiedades de programas lógicos. La lógica NPL se define en la sección 3.2 y está inspirada tanto en lo realizado para programación lógica (confrontar (29)) en cuanto a la definición de programas (aunque los nuestros generalicen a los programas de Horn) e intenciones, como en la lógica NDL del capítulo 2. De esta última tomamos la idea del trabajo en estructuras de tres géneros, que incluye una codificación interna de sucesiones de datos y una estructura temporal. Mientras que en NDL el papel de las "trazas internas" era codificar sucesiones de estados que representasen cómputos de programas itera-

tivos, aquí se tratara de representar sucesiones de fórmulas que representen deducciones, las cuales imaginaremos como cómputos de programas lógicos.

La lógica NPL es completa con el cálculo LNP (la definición del cálculo y el teorema de completitud pueden encontrarse en 3.2).

La sección 3.3 se ocupa de comparar la semántica de NPL con la semántica habitual de programas lógicos. Considerados como conjuntos de fórmulas de primer orden, estos tienen una semántica declarativa natural heredada de la lógica, y es posible dotarlos también de una semántica de punto fijo y de una semántica procedural inducida por una variante especial del procedimiento de resolución. Se conocen resultados que establecen la equivalencia entre estos tres tipos de semántica para programas de Horn, sobre interpretaciones de Herbrand (cfr.(3) y (11)).

Nosotros estableceremos la equivalencia entre semántica declarativa y semántica de punto fijo para una clase de ppogramas más amplia que los programas de Horn y sobre las estructuras standard, que incluyen en particular a las de Herbrand. A fin de relacionar estas dos semánticas con la semántica de NPL definiremos las estructuras admisibles (vistas en 2,5) y demostraremos que la semántica de NPL para programas lógicos en estructuras admisibles es equivalente a una semántica de mínimo punto fijo definible (similar a la de Cart-

wright) y coincide tambien con la semántica declarativa sobre aquellas estructuras admisibles que sean standard.

La sección 2.4 discute la potencia de NPL. Allí se demuestra que los programas regulares y los programas recursivos pueden ser simulados como programas lógicos, englobando así en nuestro formalismo los resultados de Andréka, Németi, Sain y Cartwright que se discutieron en el capítulo precedente.

Además se desarrollan una serie de aplicaciones prácticas del uso de esta lógica para razonar acerca de propiedades de programas lógicos y de estructuras de datos manipuladas por ellos.

3.2.- Definición de NPL. El cálculo LNP. Teorema de completitud.

Definición 3.2.1.- Tipo de similaridad td. Estructuras de tipo td

td es un tipo de similaridad de tres géneros t, d, s (t se le llama tiempo, d datos y s sucesiones)

Los símbolos de td son:

- Los símbolos 0, 1, + de la aritmética de Peano son del género t, y forman el tipo de similaridad t del tiempo.
- Los símbolos de d, tipo de similaridad de los datos, son del género d. Se supone que 0, 1, + no aparecen en d
- ext es un símbolo de función con dos argumentos de género s y t y valor de género d, y se supone que no aparece ni en t ni en s.

Las variables del lenguaje son $V_t = \{ i,j,k,\dots \}$ del género t, $V_d = \{x,y,z,\dots\}$ del género d y $V_s = \{u,v,w,\dots\}$ del genéro s.

De la forma habitual se definen los términos de t, d, s.

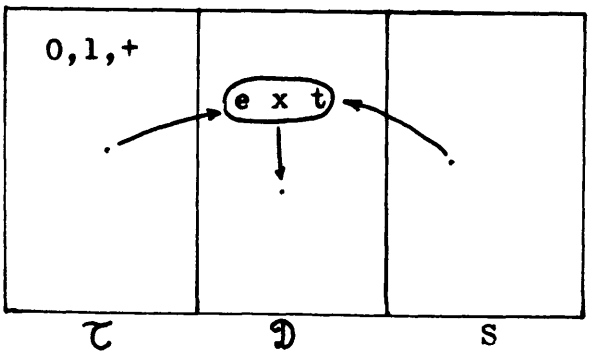
F_{td} son las fórmulas de primer orden del tipo td (donde $\tau_1 \doteq \tau_2$ solo se admite como fórmula si τ_1 y τ_2 son del mismo género).

Una estructura de tipo td será:

$$\mathfrak{M} = (\mathfrak{C}, \mathfrak{D}, S, ext^{\mathfrak{M}})$$

con $\mathfrak{C}$ estructura del tipo de similaridad t, $\mathfrak{D}$ estructura del tipo de similaridad d, S es un conjunto no vacio y $ext^{\mathfrak{M}} : S \times T \longrightarrow D$ (donde T es el universo de $\mathfrak{C}$ y D es el universo de $\mathfrak{D}$).

géneros	t tiempo	d datos	s secuencias
símbolos	0,1,+	símbolos de d	
	e x t		
variables	i,j,k,...	x,y,z,...	u,v,w,...



Notación y abreviaturas:

F_d son las fórmulas de primer orden del tipo de similaridad d

F_t son las fórmulas de primer orden del tipo de similaridad t

$i < j$ se define como $\exists k\ (\ i + (k + 1) \doteq j)$

$i \leqslant j$ se define como $(i < j \vee i \doteq j)$

$u(i) = ext(u,i)$

R_d son los símbolos de predicado del tipo de similaridad d

T_d son los términos de tipo d.

Definición 3.2.2.- Variables de programas VP

$$VP = \left\{ X_1^1,\ X_2^1,\ldots,\ X_n^1,\ldots,\ X_1^2,\ X_2^2,\ldots,\ X_n^2,\ldots,\ X_1^m,\ X_2^m,\ldots,\ X_n^m,\ldots \right\}$$

Las variables de programas representan predicados sobre D. X_n^m es un predicado de m argumentos $X_n^m \subseteq \mathcal{P}(D^m)$.

Para simplificar la notación, y si no hay posibilidad de error, usaremos X, Y, Z para referirnos a variables de programas.

Definición 3.2.3.- Atomos

$$\text{Atomos} \begin{cases} \text{Primitivos} \begin{cases} Q\tau_1,\ldots,\tau_m & \tau_1,\ldots,\tau_m \in T_d \\ \tau_1 \doteq \tau_2 & \\ \neg Q\tau_1,\ldots,\tau_m & Q \in R_d \\ \neg\, \tau_1 \doteq \tau_2 & \end{cases} \\ \text{No primitivos} \quad X_n^m \tau_1,\ldots,\tau_m \quad X_n^m \in VP, \ \tau_1,\ldots,\tau_m \in T_d \end{cases}$$

Definición 3.2.4.- CD_d^o, PL_d^o, LPF_d^o

CD_d^o Cláusulas definidas simples

$$\left.\begin{array}{l} \alpha \text{ átomo no primitivo} \\ \text{Para todo } j \ 1 \leq j \leq q, \ q \geq 0 \ \ \beta_j \text{ átomo} \end{array}\right\} \Longrightarrow \quad \alpha \longleftarrow \beta_1,\ldots,\beta_q \in CD_d^o$$

PL_d^o Programas lógicos simples

$$C_1,\ldots,C_k \in CD_d^o \Longrightarrow \{C_1,\ldots,C_k\} \in PL_d^o$$

LPF_d^o Fórmulas de programas lógicos simples

$$\varphi \in F_{td} \Longrightarrow \varphi \in LPF_d^o$$

$$\Pi \in PL_d^o \text{ y } X_n^m \in VP, \ \tau_1,\ldots,\tau_m \in T_d \Longrightarrow \Pi\, X_n^m(\tau_1,\ldots,\tau_m) \in LPF_d^o$$

$$\left.\begin{array}{l} \varphi, \psi \in LPF_d^o \\ i \in V_t, x \in V_d, u \in V_s \end{array}\right\} \Longrightarrow \neg\varphi, \ \varphi \wedge \psi, \ \exists i\varphi, \ \exists x\varphi, \ \exists u\varphi \in LPF_d^o$$

Definición 3.2.5.- CD_d, PL_d, LPF_d

CD_d Claúsulas definidas

α átomo no primitivo

Para todo j $1 \leq j \leq q$

$q \geq 0$

$$\left.\begin{array}{l} \beta_j \text{ átomo o} \\ \beta_j = \Pi X_n^m \tau_1, \ldots, \tau_m \text{ o} \\ \beta_j = \neg \Pi X_n^m \tau_1, \ldots, \tau_m \end{array}\right\} \Longrightarrow \alpha \longleftarrow \beta_1, \ldots, \beta_q \in C$$

$$\Pi \in PL_d,\ \tau_1, \ldots, \tau_m \in T_d,\ X_n^m \in VP$$

PL_d Programas lógicos

$$C_1, \ldots, C_k \in CD_d \Longrightarrow \{C_1, \ldots, C_k\} \in PL_d$$

LPF_d Fórmulas de programas lógicos

$$\varphi \in F_{td} \Longrightarrow \varphi \in LPF_d$$

$$\Pi \in PL_d \text{ y } X_n^m \in VP,\ \tau_1, \ldots, \tau_m \in T_d \Longrightarrow \Pi X_n^m(\tau_1, \ldots, \tau_m) \in LPF_d$$

$$\left.\begin{array}{l} \varphi, \psi \in LPF_d \\ i \in V_t, x \in V_d, u \in V_s \end{array}\right\} \Longrightarrow \neg\varphi,\ \varphi \wedge \psi,\ \exists i \varphi,\ \exists x \varphi,\ \exists u \varphi \in LPF_d$$

Estos programas suponen una generalización de los programas de Horn para programación lógica en estructuras de Herbrand (cfr. (29))

La idea de los programas es bastante habitual en programación. Hay procedimientos internos (las claúsulas) y procedimientos externos (los programas que aparecen en las claúsulas), Los procedimientos internos dan un resultado (el átomo α) si se cumplen todos los β_j Si los β_j son no primitivos supone una llamada a otro procedimiento interno del mismo programa. Si β_j es $(\neg)\Pi X_n^m \tau_1, \ldots, \tau_m$ estamos haciendo una llamada a un procedimiento externo: otro programa.

No todos estos programas son ejecutables en la práctica, pero los programas lógicos simples (que incluyen a los programas de Horn) si pueden, en principio, imaginarse como ejecutables.

Definición 3.2.6.-

Una aparición de una variable $x \in V_d$ en una fórmula φ se dice libre si no está dentro de una subfórmula $\exists x \varphi_o$ de φ. Análogas definiciones se tienen para $i \in V_t$, $u \in V_s$.

Definición 3.2.7.-

Sea $\Pi \in PL_d$

- V_Π es el conjunto de las variables de programas que aparecen en Π
- grado(Π) = $\max\{m / X_n^m \in V_\Pi\}$
- P_Π es el conjunto de programas de PL_d que aparecen en el lado derecho de alguna claúsula de Π

En lo que sigue asumimos que en T_d existe una sucesión infinita de términos cerrados diferentes que llamaremos $\underline{0}$, $\underline{1}$, $\underline{2}$,....

Definición 3.2.8.- Fórmula demostración$_\Pi$

Sea Π un programa lógico, grado(Π) = r

$$\text{demostración}_\Pi(u_1,\ldots,u_r,v,w,i) = \forall j(j \leq i \longrightarrow \bigvee_{C \in \Pi} \gamma_C(u_1,\ldots,u_r,v,w,j))$$

donde si C es la claúsula definida de Π $X_n^m(\tau_1,\ldots,\tau_m) \leftarrow \beta_1,\ldots,\beta_q$ y $\bar{x}$ engloba todas las variables que aparezcan libres en el lado derecho de C

$$\gamma_C(u_1,\ldots,u_m,v,w,j)=(v(j)\doteq \underline{n} \wedge w(j)\doteq \underline{m} \wedge \exists \bar{x}(\beta_1^* \wedge \ldots \wedge \beta_q^* \wedge u_1(j)\doteq \tau_1 \wedge \ldots \wedge u_m(j)\doteq \tau_m))$$

con:

si β_j átomo primitivo o $\beta_j=(\neg)\Pi'\alpha$ $\quad\beta_j^*=\beta_j$

si β_j átomo no primitivo $\beta_j= X_{n'}^{m'}(\sigma_1,\ldots,\sigma_{m'})$

$\beta_j^*= \exists j'(j'< j \wedge v(j')\doteq \underline{n'} \wedge w(j')\doteq \underline{m'} \wedge u_1(j')\doteq \sigma_1 \wedge \ldots \wedge u_{m'}(j')\doteq \sigma_{m'})$

Ejemplo:

$\mathfrak{D} = (N,\ 0^N,\ suc^N)$

$\Pi = \{\ X_1^1(0) \longleftarrow\ ,$

$X_1^1(suc(suc(x))) \longleftarrow X_1^1(x)\ \}$

<u>demostración</u>$_\Pi$ $(u,v,w,i)= \forall j(j \leq i \longrightarrow ((v(j)\doteq \underline{1} \wedge w(j)\doteq \underline{1} \wedge u(j)\doteq 0) \vee$

$(v(j)\doteq \underline{1} \wedge w(j)\doteq \underline{1} \wedge \exists x(u(j)\doteq suc(suc(x)) \wedge \exists j'(j'< j \wedge v(j')\doteq \underline{1} \wedge w(j')\doteq \underline{1}$

$\wedge\ u(j')\doteq x))))$

<u>Definición 3.2.9.</u>- Estados sobre $\mathfrak{M}$: W_{td}

Sea $\mathfrak{M} = (\mathcal{T}, \mathfrak{D}, S, ext^{\mathfrak{M}})$ estructura de tipo td. Si T es el universo de $\mathcal{T}$, D universo de $\mathfrak{D}$

$W_{td} = T^{V_t} \times D^{V_d} \times S^{V_s}$ es el conjunto de estados sobre $\mathfrak{M}$.

Cada estado $w \in W_{td}$ es una aplicación de las variables de V_t, V_d y V_s en T, D y S.

<u>Definición 3.2.10.</u>- Validez de fórmulas de LPF_d en estructuras $\mathfrak{M}$.

Sea $\mathfrak{M}$ estructura de tipo td, $\varphi \in LPF_d$ y $w \in W_{td}$

$\mathfrak{M} \models_w \varphi$ se define

· $\varphi \in F_{td}$ $\quad\mathfrak{M} \models_w \varphi$ se define del modo habitual

· $\mathcal{M} \models_w \Pi X^m_n \tau_1, \ldots, \tau_m$ $\quad r = \text{grado}(\Pi) \iff$

$\mathcal{M} \models_w \exists u_1, \ldots, u_r, v, w, i (\underline{\text{demostración}}_\Pi (u_1, \ldots, u_r, v, w, i) \wedge v(i) \doteq \underline{n} \wedge$

$w(i) \doteq \underline{m} \wedge u_1(i) \doteq \tau_1 \wedge \ldots \wedge u_m(i) \doteq \tau_m)$

· $\mathcal{M} \models_w \neg \varphi$, $\mathcal{M} \models_w \varphi \wedge \psi$, $\mathcal{M} \models_w \exists i \varphi$, $\mathcal{M} \models_w \exists x \varphi$, $\mathcal{M} \models_w \exists u \varphi$ se definen del modo habitual.

φ es válida en $\mathcal{M}$ $\quad \mathcal{M} \models \varphi \iff$ para todo $w \in W_{td}$ $\quad \mathcal{M} \models_w \varphi$

Las definiciones de consecuencia, satisfactibilidad,..son las habituales.

De esta manera queda definida la lógica NPL (Lógica no standard para programas lógicos).

Como en el capítulo anterior el término no standard hace referencia tanto a la semántica de los programas lógicos, que no es la habitual (cfr. (29) y la discusión en la sección 3.3), como a las estructuras con las que trabajamos. Tanto la estructura temporal τ como la estructura de datos $\mathfrak{D}$ pueden ser cualquier modelo de su tipo de similaridad.

En el resto de la sección propondremos un cálculo para NPL y probaremos su completitud. La definición del cálculo LNP es bastante evidente, ya que hemos definido la semántica de la fórmula $\Pi X(\tau_1, \ldots, \tau_m)$ con una fórmula de LPF_d. La completitud de LNP se obtiene con un teorema de expresividad similar a los vistos en los capítulos 1 y 2.

Definición 3.2.11.- Cálculo LNP

Axiomas

(T_o) Todas las instancias de tautologías del cálculo proposicional

resultantes de sustituir consistentemente variables proposicionales por fórmulas de LPF_d

(T_1) Un conjunto de axiomas para F_{td} de manera que junto con las reglas Modus Ponens (MP) y Generalización (G) formen un sistema de inferencia completo para F_{td}.

(Π) $\Pi\, X_n^m(x_1,\ldots,x_m) \longleftrightarrow \exists u_1,\ldots,u_r,v,w,i(\underline{\text{demostración}}_\pi(u_1,..,u_r,v,w,i)$

$\wedge\ v(i) \doteq \underline{n} \wedge w(i) \doteq \underline{m} \wedge u_1(i) \doteq x_1 \wedge \ldots \wedge u_m(i) \doteq x_m)$

$r = \text{grado}(\Pi)$

Reglas

(MP) Modus Ponens $\dfrac{\varphi\ ,\ \varphi \longrightarrow \psi}{\psi}$

(G) Generalización $\dfrac{\varphi \longrightarrow \psi}{\exists i\,\varphi \longrightarrow \exists i\,\psi}, \dfrac{\varphi \longrightarrow \psi}{\exists x\,\varphi \longrightarrow \exists x\psi}, \dfrac{\varphi \longrightarrow \psi}{\exists u\,\varphi \longrightarrow \exists u\ \psi}$

Teorema 3.2.1.- LNP es correcto

Sean $\Phi \subseteq LPF_d$, $\varphi \in LPF_d$

$$\Phi \vdash^{LNP} \varphi \implies \Phi \models \varphi$$

Demostración.-

Evidentemente cada axioma de LNP es válido en cualquier estructura $\mathcal{M}$ (el axioma (π) es la definición de validez de fórmulas con programas).

También es inmediato que si las premisas de las reglas son válidas en $\mathcal{M}$ también lo es la conclusión.

Por tanto si $\Phi \vdash^{LNP} \varphi$ tenemos $\Phi \models \varphi$

Teorema 3.2.2.-

Para cualquier $\varphi \in LPF_d$ puede construirse $\varphi' \in F_{td}$ con

$$\vdash^{LNP} (\varphi \longleftrightarrow \varphi')$$

Demostración.-

Por inducción sobre la estructura de φ

· Si $\varphi \in F_{td}$, tomamos $\varphi' = \varphi$ y (T_o) $\vdash^{LNP} (\varphi \longleftrightarrow \varphi')$

· Si $\varphi = \neg\psi$ por hipótesis de inducción $\vdash^{LNP} (\psi \longleftrightarrow \psi')$ con $\psi' \in F_{td}$. Tomando $\varphi' = \neg\psi'$ $(T_o),(MP)$ $\vdash^{LNP} (\varphi \longleftrightarrow \varphi')$

· Si $\varphi = \psi \wedge \chi$ por hipótesis de inducción $\vdash^{LNP} (\psi \longleftrightarrow \psi')$ $\psi' \in F_{td}$

$\vdash^{LNP} (\chi \longleftrightarrow \chi')$ $\chi' \in F_{td}$

Tomando $\varphi' = \psi' \wedge \chi'$ $(T_o),(MP)$ $\vdash^{LNP} (\varphi \longleftrightarrow \varphi')$

· Si $\varphi = \exists i \psi$ por hipótesis de inducción $\vdash^{LNP} (\psi \longleftrightarrow \psi')$ $\psi' \in F_{td}$

Tomando $\varphi' = \exists i \psi'$ $(T_o),(MP),(G)$ $\vdash^{LNP} (\varphi \longleftrightarrow \varphi')$

Análogamente se prueban los casos $\varphi = \exists x \psi$ o $\varphi = \exists u \psi$

· Si $\varphi = \Pi X^m_n(\tau_1,\ldots,\tau_m)$ Consideremos la fórmula

$\underline{\text{demostración}}_\Pi (u_1,\ldots,u_r,v,w,i) = \forall j (j \leq i \longrightarrow \bigvee_{C \in \Pi} \gamma_C(u_1,\ldots,u_r,v,w,j)$

donde si $C = X^{m'}_{n'}(\sigma_1,..,\sigma_{m'}) \longleftarrow \beta_1,..,\beta_q$ y $\bar{x}$ engloba todas las variables libres del lado derecho de C

$\gamma_C = v(j) \doteq \underline{n}' \wedge w(j) \doteq \underline{m}' \wedge \exists \bar{x} (\beta^*_1 \wedge \ldots \wedge \beta^*_q \wedge u_1(j) \doteq \sigma_1 \wedge \ldots \wedge u_{m'} \doteq \sigma_{m'}$

con $\beta^*_j = \beta_j$ si $\beta_j = (\neg)\Pi' \propto$ $\propto$ átomo no primitivo.

Por hipótesis de inducción para estos β_j existe $\beta'_j \in F_{td}$ con

$$\vdash^{LNP} (\beta_j \longleftrightarrow \beta'_j)$$

Tomando $\beta'_j = \beta^*_j$ si β_j átomo tenemos $(T_o),(MP)$ $\vdash^{LNP} (\gamma_C \longleftrightarrow \gamma'_C)$ c

$\gamma'_C = v(j) \doteq \underline{n}' \wedge w(j) \doteq \underline{m}' \wedge \exists \bar{x}(\beta'_1 \wedge \ldots \wedge \beta'_q \wedge u_1(j) \doteq \sigma_1 \wedge \ldots \wedge u_{m'}(j) \doteq \sigma_m$

y $\gamma'_C \in F_{td}$

Si llamamos $\underline{\text{demostración}}'_\pi = \forall j(j \leq i \longrightarrow \bigvee_{C \in \pi} \gamma'_C) \in F_{td}$ y obtenemos

$(T_o),(MP),(G) \quad \vdash^{LNP} (\underline{\text{demostracion}}_\pi \longleftrightarrow \underline{\text{demostracion}}'_\pi)$

Tomando $\varphi' = \exists u_1,\ldots,u_r,v,w,i(\underline{\text{demostracion}}'_\pi (u_1,\ldots,u_r,v,w,i) \wedge$

$v(i) \doteq \underline{n} \wedge w(i) \doteq \underline{m} \wedge u_1(i) \doteq \tau_1 \wedge \ldots \wedge u_m(i) \doteq \tau_m)$

concluimos $(\Pi),(T_o),(MP),(G) \quad \vdash^{LNP} (\Pi\, X^m_n(\tau_1,\ldots,\tau_m) \longleftrightarrow \varphi')$

Teorema 3.2.3.- Completitud de LNP

Para cualquier $\Phi \subseteq LPF_d$, $\varphi \in LPF_d$

$$\Phi \models \varphi \quad \Longrightarrow \quad \Phi \vdash^{LNP} \varphi$$

Demostración.-

Es inmediata con el teorema 3.2.2. Para φ existe $\varphi' \in F_{td}$ con $\vdash^{LNP} (\varphi \longleftrightarrow \varphi')$. Como $\Phi \models \varphi$ y LNP es correcto $\Phi \models \varphi'$

Dado que $(T_1),(MP),(G)$ forman un sistema completo para F_{td} tenemos

$\Phi \vdash^{LNP} \varphi'$ y con $(T_o),(MP)$ $\Phi \vdash^{LNP} \varphi$

Obsérvese que, de nuevo, el paso clave de la demostración de la completitud del cálculo es el teorema 3.2.2, un teorema de expresividad similar a los vistos en capítulos anteriores.

Ejemplos del uso de LNP:

El cálculo LNP puede ser utilizado en algunos casos para demostrar la equivalencia de programas lógicos.

Tomemos $\mathfrak{D} = (N, 0^N, suc^N)$ y los programas

$$\Pi_1 = \{ X_1^1(0) \leftarrow \quad , \quad X_1^1(suc(suc(x))) \leftarrow X_1^1(x) \}$$

$$\Pi_2 = \{ X_1^1(0) \leftarrow \quad , \quad X_1^1(suc(suc(0))) \leftarrow X_1^1(0) \quad , \quad X_1^1(suc(suc(suc(suc(0))))) \leftarrow X_1^1(suc(suc(x))) \}$$

Intuitivamente tanto Π_1 como Π_2 definen el conjunto de los números pares. Por tanto parecen equivalentes y de hecho se tiene que

$$\vdash^{LNP} \forall x(\Pi_1 X_1^1(x) \leftrightarrow \Pi_2 X_1^1(x))$$

Según el teorema de completitud para LNP basta demostrar que

$\mathcal{M} \models \forall x(\Pi_1 X_1^1(x) \leftrightarrow \Pi_2 X_1^1(x))$ cualquiera que sea la estructura $\mathcal{M}$ de tipo td.

Pero ésto es consecuencia de que dada $\mathcal{M}$ y fijados $u,v,w \in S$, $i \in T$ en $\mathcal{M}$ se tiene

$$\mathcal{M} \models \underline{\text{demostración}}_{\Pi_1}(u,v,w,i) \iff \mathcal{M} \models \underline{\text{demostración}}_{\Pi_2}(u,v,w,i)$$

lo cual es una simple consecuencia de la forma de las fórmulas $\underline{\text{demostración}}_{\Pi}$ y del hecho de que dos objetos de $\mathfrak{D}$ de las formas suc(suc(y)), suc(suc(suc(suc(y)))) son en particular de las formas x, suc(suc(x)).

Nótese que la fórmula afirma que si Π_1 computa $X_1^1(x)$ también lo computa Π_2 (y viceversa), pero no afirma que Π_1 pueda computar algo.

Desgraciadamente, el conjunto de resultados posibles utilizando

LNP y apoyandose en el conjunto vacio de hipótesis no es muy amplio. Esto es debido a la falta de información sobre el conjunto de secuencias S y la estructura temporal τ .

Por ejemplo consideremos el programa

$$\Pi_3 = \{\ X_1^1(0) \leftarrow \quad ,$$

$$X_1^1(\mathrm{suc}(\mathrm{suc}(0))) \leftarrow \quad ,$$

$$X_1^1(\mathrm{suc}(\mathrm{suc}(\mathrm{suc}(\mathrm{suc}(0))))) \leftarrow X_1^1(\mathrm{suc}(\mathrm{suc}(0)))\ \}$$

Intuitivamente Π_3 sigue siendo equivalente a Π_1 y a Π_2, pero las fórmulas que expresan las equivalencias

$$\forall x(\Pi_1 X_1^1(x) \leftrightarrow \Pi_3 X_1^1(x)) \qquad \forall x(\Pi_2 X_1^1(x) \leftrightarrow \Pi_3 X_1^1(x))$$

ya no son lógicamente válidas en NPL.

Para comprobarlo consideremos la siguiente estructura de tipo td

$$\mathcal{M} = (\tau, \mathfrak{D}, S, \mathrm{ext}^{\mathcal{M}})$$

con $\tau = (\mathbb{N}, 0^{\mathbb{N}}, 1^{\mathbb{N}}, +^{\mathbb{N}})$

$\mathfrak{D} = (\mathbb{N}, 0'^{\mathbb{N}}, \mathrm{suc}^{\mathbb{N}})$

$S = \{a \in \mathbb{N}^{\omega} / a_0 \neq 0'\}$

$\mathrm{ext}^{\mathcal{M}}(a,i) = a_i = a(i)$

S son todas las sucesiones de números naturales que no empiezan por 0'

Si u(0) no puede ser 0', para cualquier $u \in S$ tenemos

$$\mathcal{M} \models \forall x \neg \Pi_1 X_1^1(x)$$

y sin embargo

$$\mathcal{M} \models \Pi_3 X_1^1(\mathrm{suc}(\mathrm{suc}(0')))$$

cogiendo $u \in S$ con $u(0) = \mathrm{suc}(\mathrm{suc}(0'))$, v y w con $v(0) = \underline{1}$ y $w(0) = \underline{1}$

tenemos inmediatamente

$$\mathcal{M} \models \underline{\text{demostración}}_{\pi_3}(u,v,w,0)$$

En la sección 3.3 estudiaremos como fortalecer nuestro sistema de inferencia, exigiendo como axiomas ciertas propiedades para τ, S y ext, análogamente como sucedía en 2.4. Basándonos en estos axiomas LNP sí será capaz de inferir propiedades interesantes acerca de programas lógicos.

3.3.- Estructuras standard y estructuras admisibles. Semántica de programas lógicos simples sobre estructuras standard y estructuras admisibles.

En esta sección discutiremos y compararemos las distintas semánticas propuestas para programas lógicos.

Trabajaremos con los programas lógicos simples (una clase mas amplia que los programas de Horn) y sobre estructuras standard en las cuales la estructura de los datos es accesible (todo elemento del dominio es el valor de un término cerrado). Estas estructuras incluyen a las interpretaciones de Herbrand.

Los programas lógicos simples pueden considerarse como un conjunto de fórmulas de primer orden, lo que supone una semántica declarativa natural, heredada de la lógica. También se les puede dotar de una semántica de punto fijo. Para programas de Horn es conocida la equivalencia de ambas semánticas sobre interpretaciones de Herbrand. Nosotros generalizaremos este resultado a los programas lógicos simples sobre estructuras standard.

Además, probaremos que la semántica de NPL para programas lógicos en ciertas estructuras (las estructuras admisibles, ya definidas en 2.5) es equivalente a una semántica de mínimo punto fijo definible, (con similitudes con los resultados de Cartwright comentados en 2.5) y coincide con la semántica declarativa para las estructuras admisi-

bles y standard.

Definición 3.3.1.- Tipos de datos accesibles y sus diagramas

Un tipo de datos accesible es cualquier estructura $\mathfrak{D}$ de tipo de similaridad d, tal que todo elemento del dominio de $\mathfrak{D}$ es el valor en $\mathfrak{D}$ de un término cerrado del lenguaje de $\mathfrak{D}$.

Para un tipo de datos accesible $\mathfrak{D}$, el diagrama de $\mathfrak{D}$, $\Delta_{\mathfrak{D}}$, es el conjunto de sentencias atómicas del tipo $\tau \doteq \sigma$, $\neg\, \tau \doteq \sigma$, $Q(\tau_1,..,\tau_n)$ $\neg\, Q(\tau_1,..,\tau_n)$del lenguaje de $\mathfrak{D}$, verdaderas en $\mathfrak{D}$.

Proposición 3.3.1.-

Sea $\mathfrak{D}$ un tipo de datos accesible y $\mathfrak{D}'$ una estructura del mismo tipo de similaridad que $\mathfrak{D}$

$$\mathfrak{D} \subseteq \mathfrak{D}' \iff \mathfrak{D}' \models \Delta_{\mathfrak{D}}$$

y $\mathfrak{D}' \models \Delta_{\mathfrak{D}}$ y $\mathfrak{D}'$ accesible $\Longrightarrow$ $\mathfrak{D}' \simeq \mathfrak{D}$

($\mathfrak{D} \subseteq \mathfrak{D}'$ indica que $\mathfrak{D}$ es, salvo isomorfismo, subestructura de $\mathfrak{D}'$)

Definición 3.3.2.-

Sea $\mathfrak{D}$ estructura de tipo de similaridad d. $\mathcal{M}_{\mathfrak{D}} = (\mathcal{T} , \mathfrak{D} , S, ext^{\mathcal{M}_{\mathfrak{D}}})$ es la estructura standard de tipo td sobre $\mathfrak{D}$ si

(1) $\mathcal{T} = (N, 0^N, 1^N, +^N)$

(2) $S = D^+ = \bigcup_{n \in N-\{0\}} D^n$

(3) Si $s \in D^n$ $ext^{\mathcal{M}_{\mathfrak{D}}}(s,i) = \begin{cases} s(i) = s_i & \text{si } i \leq n \\ s(n) = s_n & \text{en otro caso} \end{cases}$

Semántica de programas lógicos simples sobre structuras standard.

Cada variable $X \in VP$ se puede interpretar como un nuevo predicado sobre $\mathcal{D}$. Un programa lógico Π puede entenderse como una definición implícita de nuevos predicados sobre $\mathcal{D}$ (las variables de programas $X \in VP$).

Los programas lógicos simples pueden, entonces, representar fórmulas de primer orden:

$$\alpha \leftarrow \beta_1, \ldots, \beta_q \quad : \quad (\beta_1 \wedge \ldots \wedge \beta_q \rightarrow \alpha)^{\forall}$$

$$\alpha \leftarrow \quad : \quad (\alpha)^{\forall}$$

donde $(\ )^{\forall}$ representa el cierre universal de todas las variables libres.

Estas fórmulas, asociadas a cada programa lógico simple, dan el significado lógico de Π en $\mathcal{D}$ ampliada con las interpretaciones para los elementos de V_π.

Definición 3.3.3.- $B_\pi^{\mathcal{D}}$, $\mathcal{P}^{\mathcal{D}}(B_\pi^{\mathcal{D}})$

Sea $\mathcal{D}$ un tipo de datos accesible y Π un programa lógico simple. La base de Herbrand de Π sobre $\mathcal{D}$ ($B_\pi^{\mathcal{D}}$) son todos los átomos no primitivos $X(\tau_1, \ldots, \tau_m)$ donde $X \in V_\pi$ y $\tau_1, \ldots, \tau_m$ son términos cerrados de T_d.

$I \subseteq B_\pi^{\mathcal{D}}$ es $\mathcal{D}$-cerrado si y solo si $X(\sigma_1, \ldots, \sigma_m) \in I$ si $X(\tau_1, \ldots, \tau_m) \in I$ y $\sigma_1^{\mathcal{D}} = \tau_1^{\mathcal{D}}$ y .... y $\sigma_m^{\mathcal{D}} = \tau_m^{\mathcal{D}}$

$$\mathcal{P}^{\mathcal{D}}(B_\pi^{\mathcal{D}}) = \{ I \ / \ I \in B_\pi^{\mathcal{D}} \ , \ I \text{ es } \mathcal{D}\text{-cerrado} \}$$

$I \in \mathcal{P}^{\mathcal{D}}(B_\pi^{\mathcal{D}})$ se le llama interpretación libre de Π sobre $\mathcal{D}$.

Los elementos de $\mathcal{P}^{\mathcal{D}}$ ($B_{\pi}^{\mathcal{D}}$) pueden identificarse con las estructuras que amplían a $\mathcal{D}$ dando una interpretadión para cada X de V_{π}

Definición 3.3.4.-

C' es una instancia básica sobre $\mathcal{D}$ de una claúsula C de π si C' resulta de C sustituyendo las variables por términos cerrados de T_d

I, interpretación libre de π sobre $\mathcal{D}$, es un modelo de C' ($I \models C'$) $C' = \alpha' \leftarrow \beta_1', \ldots, \beta_q'$ si y solo si $I \models \beta_1' \wedge \ldots \wedge \beta_q' \Longrightarrow I \models \alpha'$ (donde $I \models \alpha' \iff \alpha' \in I$ y $I \models \beta_1' \wedge \ldots \wedge \beta_q' \iff$ para todo j, $1 \leq j \leq q$ $\beta_j' \in \Delta_{\mathcal{D}} \cup I$)

I, interpretación libre de π sobre $\mathcal{D}$, es un modelo de C ($I \models C$) si y solo si I es modelo de C' para cada instancia básica C' de C sobre $\mathcal{D}$.

Una interpretación libre I de π sobre $\mathcal{D}$ es un modelo libre de π sobre $\mathcal{D}$ ($I \models \pi$) si y solo si I es modelo de cada claúsula C de π

Proposición 3.3.2.-

$\mathcal{P}^{\mathcal{D}}(B_{\pi}^{\mathcal{D}})$ es un retículo completo con respecto al orden parcial $\subseteq$

Demostración.-

La cota superior mínima se obtiene con la $\cup$ conjuntista y la cota inferior máxima con la $\cap$ conjuntista. Claramente las uniones (finitas o infinitas) de partes $\mathcal{D}$-cerradas de B_{π} son $\mathcal{D}$-cerradas.

El elemento mínimo es $\varnothing$ y el máximo B_{π} (trivialmente $\mathcal{D}$-cerra

dos).

Además si $(I_i)_{i\in M}$ es una colección de elementos de $\mathcal{P}^{\mathcal{D}}(B_\pi^{\mathcal{D}})$, $(I_i)_{i\in M}$ tiene cota superior mínima (cota inferior máxima) en $\mathcal{P}^{\mathcal{D}}(B_\pi^{\mathcal{D}})$ que se obtiene con $\bigcup_{i\in M} I_i$ ($\bigcap_{i\in M} I_i$)

Proposición 3.3.3.- (Lema de intersección de modelos)

Sea Π un programa lógico simple. Si $(I_i)_{i\in M}$ es un conjunto no vacío de modelos libres de Π sobre $\mathcal{D}$ entonces $\bigcap_{i\in M} I_i$ es un modelo libre de Π sobre $\mathcal{D}$.

Demostración.-

Por la proposición 3.3.2 como cada $I_i \in \mathcal{P}^{\mathcal{D}}(B_\pi^{\mathcal{D}})$ tenemos $\bigcap_{i\in M} I_i \in \mathcal{P}^{\mathcal{D}}(B_\pi^{\mathcal{D}})$

Sea $\alpha \leftarrow \beta_1, \ldots, \beta_q$ instancia básica de una claúsula definida de Π

Si para todo j, $1 \leq j \leq q$ $\begin{cases} \beta_j \text{ átomo primitivo} \Rightarrow \beta_j \in \Delta_{\mathcal{D}} \\ \beta_j \text{ átomo no primitivo} \Rightarrow \beta_j \in \bigcap I_i \Rightarrow \\ \text{para todo i } \beta_j \in I_i \end{cases}$

Como cada I_i es un modelo libre de Π sobre $\mathcal{D}$ tenemos que para todo i $\alpha \in I_i \Longrightarrow \alpha \in \bigcap I_i$

Luego $\bigcap_{i\in M} I_i$ es un modelo libre de Π sobre $\mathcal{D}$.

Si Π es un programa lógico simple denominaremos $I_\pi^{\mathcal{D}}$ a la intersección de todos los modelos libres de Π sobre $\mathcal{D}$.

Como $B_\pi^{\mathcal{D}}$ es un modelo libre de Π sobre $\mathcal{D}$ la colección de modelos es no vacía, y por la proposición 3.3.3 $I_\pi^{\mathcal{D}}$ es un modelo libre de

π sobre $\mathcal{D}$. Con ello $I_\pi^{\mathcal{D}}$ es el menor modelo libre de π sobre $\mathcal{D}$ (en el sentido de la relación $\subseteq$).

Definición 3.3.5.- $\bar{\pi}$, C^+, C^-

Sea C una claúsula definida sin variables $X(\tau_1,\ldots,\tau_m) \leftarrow \beta_1,\ldots,\beta_q$

$C^- = \{\beta_1,\ldots,\beta_q\}$

$C^+ = \{X(\sigma_1,\ldots,\sigma_m) / \sigma_1,\ldots,\sigma_m$ términos cerrados de Td y

$\sigma_1^{\mathcal{D}} = \tau_1^{\mathcal{D}}, \ldots, \sigma_m^{\mathcal{D}} = \tau_m^{\mathcal{D}}\}$

Sea π un programa lógico simple. $\bar{\pi}$ es el conjunto de todas las instancias básicas de claúsulas definidas de π.

Definición 3.3.6.- $T_\pi^{\mathcal{D}}$

Sea π un programa lógico simple

$T_\pi^{\mathcal{D}} : \mathcal{P}^{\mathcal{D}}(B_\pi^{\mathcal{D}}) \longrightarrow \mathcal{P}^{\mathcal{D}}(B_\pi^{\mathcal{D}})$ con

$T_\pi^{\mathcal{D}}(I) = \{\alpha \in B_\pi^{\mathcal{D}} /$ existe $C \in \bar{\pi}$ tal que $C^- \subseteq I \cup \Delta_{\mathcal{D}}$ y $\alpha \in C^+\}$

$T_\pi^{\mathcal{D}}$ esta bien definido pues si $I \in \mathcal{P}^{\mathcal{D}}(B_\pi^{\mathcal{D}})$ entonces $T_\pi^{\mathcal{D}}(I) \in B_\pi^{\mathcal{D}}$ por definición y $T_\pi^{\mathcal{D}}(I)$ es claramente $\mathcal{D}$-cerrado por la definición de C^+.

Proposición 3.3.4.-

Sea π un programa lógico simple. El operador $T_\pi^{\mathcal{D}}$ es continuo en el retículo $\mathcal{P}^{\mathcal{D}}(B_\pi^{\mathcal{D}})$

Demostración.-

Sea X un conjunto dirigido, $X \subseteq \mathcal{P}^{\mathcal{D}}(B_\pi^{\mathcal{D}})$

(1) $\bigcup T_\pi^{\mathcal{D}}(X) \subseteq T_\pi^{\mathcal{D}}(\bigcup X)$

Si $\alpha \in T_\pi^{\mathcal{D}}(X) \Longrightarrow$ existe $I \in X$ con $\alpha \in T_\pi^{\mathcal{D}}(I) \Longrightarrow$ existe $C \in \bar{\pi}$

con $\alpha \in C^+$ y $C^- \subseteq I \cup \Delta_{\mathcal{D}} \Rightarrow (I \in X)$ existe $C \in \overline{\pi}$ con $\alpha \in C^+$ y $C^- \subseteq \cup X \cup \Delta_{\mathcal{D}} \Rightarrow \alpha \in T_\pi^{\mathcal{D}}(\cup X)$

(2) $T_\pi^{\mathcal{D}}(\cup X) \subseteq \cup T_\pi^{\mathcal{D}}(X)$

Si $\alpha \in T_\pi^{\mathcal{D}}(\cup X) \Rightarrow$ existe $C \in \overline{\pi}$ con $C^- \subseteq \cup X \cup \Delta_{\mathcal{D}}$ y $\alpha \in C^+$

$C^- \subseteq \cup X \cup \Delta_{\mathcal{D}} \iff$ para todo $\beta \in C^-$ $\begin{cases} \beta \text{ primitivo} \Rightarrow \beta \in \Delta_{\mathcal{D}} \\ \beta \text{ no primitivo} \Rightarrow \\ \text{existe } I^\beta \text{ con } \beta \in I^\beta \end{cases}$

Como X es dirigido existe $I \in X$ con $I^\beta \subseteq I$ para cada β no primitivo.

Luego si $C^- \subseteq \cup X \cup \Delta_{\mathcal{D}}$ entonces $C^- \subseteq I \cup \Delta_{\mathcal{D}}$ y tenemos que si $\alpha \in T_\pi^{\mathcal{D}}(\cup X) \Rightarrow \alpha \in T_\pi^{\mathcal{D}}(I) \Rightarrow \alpha \in \cup T_\pi^{\mathcal{D}}(X)$

Proposición 3.3.5.-

Sea π un programa lógico simple.

I es un modelo libre de π sobre $\mathcal{D} \iff T_\pi^{\mathcal{D}}(I) \subseteq I$

Demostración.-

I es un modelo libre de Π sobre $\mathcal{D} \iff$ I es $\mathcal{D}$-cerrado y para cualquier $C \in \overline{\Pi}$ (para todo $\beta \in C^-$ tenemos $\beta \in I \cup \Delta_{\mathcal{D}} \Rightarrow \alpha \in I$) $\iff$ para toda $C \in \overline{\Pi}$ ($C^- \subseteq I \cup \Delta_{\mathcal{D}} \Rightarrow C^+ \subseteq I$) $\iff T_\pi^{\mathcal{D}}(I) \subseteq I$

Por la proposición 3.3.4 $T_\pi^{\mathcal{D}}$ es continuo en $\mathcal{P}^{\mathcal{D}}(B_\pi^{\mathcal{D}})$. Por el teorema de Kleene (enunciado en 2,5 para c.p.o.s) existe un mínimo punto fijo $T_\pi^{\mathcal{D}} \uparrow_\omega \in \mathcal{P}^{\mathcal{D}}(B_\pi^{\mathcal{D}})$

Teorema 3.3.1.-

Sea Π un programa lógico simple.

$$I_{\Pi}^{\mathfrak{D}} = T_{\Pi}^{\mathfrak{D}} \uparrow_{\omega}$$

Demostración.-

Probaremos que $I_{\Pi}^{\mathfrak{D}}$ es punto fijo y que es mínimo.

(1) $T_{\Pi}^{\mathfrak{D}} (I_{\Pi}^{\mathfrak{D}}) \subseteq I_{\Pi}^{\mathfrak{D}}$ pues $I_{\Pi}^{\mathfrak{D}}$ es modelo libre de Π sobre $\mathfrak{D}$ (Proposición 3.3.3 y Proposición 3.3.5)

(2) Por (1) $\{ I_{\Pi}^{\mathfrak{D}} , T_{\Pi}^{\mathfrak{D}} (I_{\Pi}^{\mathfrak{D}}) \}$ es un conjunto dirigido con cota superior mínima $I_{\Pi}^{\mathfrak{D}}$. Como $T_{\Pi}^{\mathfrak{D}}$ es continuo tenemos

$T_{\Pi}^{\mathfrak{D}} (T_{\Pi}^{\mathfrak{D}} (I_{\Pi}^{\mathfrak{D}})) \subseteq T_{\Pi}^{\mathfrak{D}} (I_{\Pi}^{\mathfrak{D}})$

Luego por la proposición 3.3.5 $T_{\Pi}^{\mathfrak{D}} (I_{\Pi}^{\mathfrak{D}})$ es un modelo libre de Π sobre $\mathfrak{D}$. Como $I_{\Pi}^{\mathfrak{D}}$ es el menor modelo libre de Π sobre $\mathfrak{D}$

$T_{\Pi}^{\mathfrak{D}} (I_{\Pi}^{\mathfrak{D}}) \supseteq I_{\Pi}^{\mathfrak{D}}$

(3) Hemos visto que $I_{\Pi}^{\mathfrak{D}}$ es punto fijo. Queda ver que es el mínimo. Si I' es punto fijo de $T_{\Pi}^{\mathfrak{D}}$ tenemos $T_{\Pi}^{\mathfrak{D}} (I') \subseteq I'$. Luego por la proposición 3.3.5 I' es modelo libre de Π sobre $\mathfrak{D}$, y por lo tanto $I_{\Pi}^{\mathfrak{D}} \subseteq I'$

Definición 3.3.7.- $\overline{C}_{\Pi}$

Sea Π un programa lógico simple.

$$\overline{C}_{\Pi} = \{ (X(\tau_1, \ldots, \tau_m) \wedge \sigma_1 \doteq \tau_1 \wedge \ldots \wedge \tau_m \doteq \sigma_m) \longrightarrow X(\sigma_1, \ldots, \sigma_m) \; / \; \tau_1, \ldots, \tau_m, \sigma_1, \ldots, \sigma_m \text{ términos cerrados de } T_d \text{ y } X \in V_{\Pi} \}$$

$\overline{C}_{\Pi}$ es un conjunto de sentencias que nos van a servir para caracterizar, en la lógica proposicional, las interpretaciones $\mathfrak{D}$-cerradas.

Proposición 3.3.6.- (Adaptación del Teorema de Herbrand).

Sea $\mathfrak{D}$ un tipo de datos accesible y sea Π un programa lógico simple. Son equivalentes:

(1) Π tiene un modelo libre sobre $\mathfrak{D}$.

(2) $\Pi \cup \Delta_{\mathfrak{D}}$ es satisfactible en el sentido de la lógica de primer orden con igualdad.

(3) $\overline{\Pi} \cup \overline{C}_{\Pi} \cup \Delta_{\mathfrak{D}}$ es satisfactible en el sentido de la lógica proposicional.

Demostración.-

(1) $\Longrightarrow$(2) Evidente, pues un modelo libre de Π sobre $\mathfrak{D}$ define una estructura $\mathfrak{D}'$ que es ampliación de $\mathfrak{D}$ con una interpretación de los X que aparecen en Π, tal que $\mathfrak{D}' \models \Pi \cup \Delta_{\mathfrak{D}}$. Esto ya se ha comentado antes.

(2) $\Longrightarrow$ (3) Sea d' el tipo de similaridad resultante de ampliar d con los símbolos de V_{Π}.

(2) significa que $\mathfrak{D}' \models \Pi \cup \Delta_{\mathfrak{D}}$ para cierta estructura $\mathfrak{D}'$ de tipo d'.

Sea $A^{\mathfrak{D}}$ la colección de todos los átomos cerrados primitivos de tipo d. Definimos $\nu : A^{\mathfrak{D}} \cup B_{\Pi}^{\mathfrak{D}} \longrightarrow \{0,1\}$ con:

$$\nu(\alpha) = \begin{cases} 1 & \text{si } \mathfrak{D}' \models \alpha \\ 0 & \text{en otro caso} \end{cases}$$

Es fácil comprobar que $\nu \models_0 \overline{\Pi} \cup \overline{C}_{\Pi} \cup \Delta_{\mathfrak{D}}$

(3) $\Longrightarrow$ (1) Suponemos que existe $\nu : A^{\mathfrak{D}} \cup B_{\Pi}^{\mathfrak{D}} \longrightarrow \{0,1\}$ tal que

$\nu \models_o \bar{\Pi} \cup \bar{C} \cup \Delta_{\mathfrak{D}}$. Sea $I \subseteq B_\Pi^{\mathfrak{D}}$ definida como

$I = \{\alpha \in B_\Pi^{\mathfrak{D}} \ / \nu(\alpha) = 1\}$

Como $\nu \models_o \bar{C}_\Pi$, I es $\mathfrak{D}$-cerrado. Como $\nu \models_o \bar{\Pi} \cup \Delta_{\mathfrak{D}}$ puede comprobarse que $I \models \Pi$.

Teorema 3.3.2.-

Sea Π un programa lógico simple.

$$I_\Pi^{\mathfrak{D}} = \{\alpha \in B_\Pi^{\mathfrak{D}} \ / \ \Pi \cup \Delta_{\mathfrak{D}} \models \alpha\}$$

Demostración.-

Para cualquier $\alpha \in B_\Pi^{\mathfrak{D}}$:

$\Pi \cup \Delta_{\mathfrak{D}} \models \alpha$

$\iff \Pi \cup \Delta_{\mathfrak{D}} \cup \{\neg\alpha\}$ es insatisfactible

$\iff$ (Proposición 3.3.6) Cualquier modelo libre de Π sobre $\mathfrak{D}$ no es modelo de $\neg\alpha$

$\iff \alpha$ es cierta en todos los modelos libres de Π sobre $\mathfrak{D}$

$\iff \alpha \in I_\Pi^{\mathfrak{D}}$

Desde un punto de vista lógico la semántica idónea para un programa lógico simple Π es la del conjunto de consecuencias lógicas de $\Pi \cup \Delta_{\mathfrak{D}}$. A esta semántica se la llama declarativa. El teorema 3.3.2 nos asegura que el conjunto de átomos que son consecuencia lógica de $\Pi \cup \Delta_{\mathfrak{D}}$ es $I_\Pi^{\mathfrak{D}}$.

Además por el teorema 3.3.1 $I_\Pi^{\mathfrak{D}} = T_\Pi^{\mathfrak{D}} \uparrow_\omega$ mínimo punto fijo

de $T_\Pi^{\mathfrak{D}}$. Por consiguiente la semántica declarativa y la de punto fijo (o denotacional) de Π sobre $\mathfrak{D}$ coinciden.

Esto supone una analogía con los programas de Horn sobre interpretaciones de Herbrand. En estas, la semántica denotacional habitual es la de mínimo punto fijo de T_Π (definido similarmente: todo esto puede encontrarse en (29)).

Más adelante probaremos que la semántica en NPL de Π sobre la estructura standard $\mathcal{M}_{\mathfrak{D}}$ (ver definición 3.3.2) coincide con la semántica de mínimo punto fijo de $T_\Pi^{\mathfrak{D}}$

Sin embargo, la semántica para programas lógicos simples del mínimo punto fijo del operador $T_\Pi^{\mathfrak{D}}$ tiene sus dificultades desde el punto de vista de la lógica NPL.

Consideremos el programa $\Pi = \{ X(0) \leftarrow \quad , \quad X(suc(x)) \leftarrow X(x) \}$

para el tipo de datos accesible $\mathfrak{D} = (N, 0^N, suc^N)$

Es fácil comprobar que la semántica de mínimo punto fijo dada por $I_\Pi^{\mathfrak{D}}$ coincide con la semántica de NPL (definición 3.2.10) en $\mathcal{M}_{\mathfrak{D}} = (\mathcal{T} , \mathfrak{D} , S, ext^{\mathcal{M}_{\mathfrak{D}}})$.

Se tiene que $\mathcal{M}_{\mathfrak{D}} \models \forall x \Pi X(x)$

Si tomamos un modelo no standard $\mathcal{M}' = (\mathcal{T}', \mathfrak{D}', S', ext'^{\mathcal{M}'})$ de $Th(\mathcal{M}_{\mathfrak{D}})$ seguirá siendo cierto que $\mathcal{M}' \models \forall x \Pi X(x)$ ya que $\mathcal{M}' \equiv \mathcal{M}$. Sin embargo, si extendemos la definición de $I_\Pi^{\mathfrak{D}}$ como

punto fijo mínimo de $T_\Pi^{\mathfrak{D}}$ a $\mathfrak{D}'$ vamos a obtener que $I_\Pi^{\mathfrak{D}}$ interpreta X en $\mathfrak{D}'$ como el conjunto de los números standard y no cumple $\forall x \Pi X(x)$.

Es decir: la semántica de mínimo punto fijo no es compatible con la lógica de primer orden, mientras que la semántica de la definición 3.3.2 sí lo es.

Vamos a ver que los dos puntos de vista pueden conciliarse si se introduce la idea de mínimo punto fijo definible.

Para ello es necesario extender la noción de interpretación libre sobre $\mathfrak{D}$ y $T_\Pi^{\mathfrak{D}}$ al caso en que $\mathfrak{D}$ es una estructura arbitraria de tipo d, no necesariamente accesible.

Definición 3.3.8.-

Sea d^+ el resultado de ampliar el tipo de similaridad d con una nueva constante $\underline{a}$ por cada elemento a del dominio D de $\mathfrak{D}$:

$\Delta_{\mathfrak{D}}$ es ahora la colección de átomos $\neg Q(\underline{a}_1,.,\underline{a}_m), Q(\underline{a}_1,\ldots,\underline{a}_m)$, $\underline{a}_1 \doteq \underline{a}_2$, $\neg \underline{a}_1 \doteq \underline{a}_2$ verdaderos en $\mathfrak{D}$ (al interpretar $\underline{a}_i$ como a_i).

La base de Herbrand $B_\Pi^{\mathfrak{D}}$ es la colección de todos los átomos $X(\underline{a}_1,\ldots,\underline{a}_m)$ donde $X \in V_\Pi$ y $a_1,\ldots,a_m \in D$.

Las interpretaciones libres de Π sobre $\mathfrak{D}$ se identifican con subconjuntos $I \subseteq B_\Pi^{\mathfrak{D}}$. Cada una de ellas corresponde a la ampliación de $\mathfrak{D}$ que interpreta cada X de V_Π como la colección de las m-tuplas

$(a_1,\ldots,a_m) \in D^m$ tales que $X(\underline{a}_1,\ldots,\underline{a}_m) \in I$

Dada $C \in \Pi$ si las variables de C se sustituyen por elementos de $\mathfrak{D}$, se evaluan todos los términos que queden, y se escribe cada valor a que resulte mediante su constante asociada $\underline{a}$, resulta una claúsula definida de la forma $\alpha \leftarrow \beta_1,\ldots,\beta_q$ donde α es de la forma $X(\underline{a}_1,\ldots,\underline{a}_m)$ y cada β_j es de la forma $X(\underline{a}_1,\ldots,\underline{a}_m)$, $(\neg)Q(\underline{a}_1,\ldots,\underline{a}_m)$, o $(\neg)\ \underline{a}_1 \doteq \underline{a}_2$. Este tipo de claúsulas diremos que son instancias básicas sobre $\mathfrak{D}$ de claúsulas de Π.

El operador $T_\Pi^{\mathfrak{D}}$ queda definido:

$T_\Pi^{\mathfrak{D}} : \mathcal{P}(B_\Pi^{\mathfrak{D}}) \longrightarrow \mathcal{P}(B_\Pi^{\mathfrak{D}})$ y

$T_\Pi^{\mathfrak{D}}(I) = \{\alpha$ / existe $\alpha \leftarrow \beta_1,\ldots,\beta_q$ instancia básica de una claúsula de Π sobre $\mathfrak{D}$ tal que $\{\beta_1,\ldots,\beta_q\} \subseteq I \cup \Delta_{\mathfrak{D}}\}$

Con estas definiciones es sencillo generalizar los resultados vistos:

· $I \subseteq B_\Pi^{\mathfrak{D}}$ es modelo libre de Π si y solo si $T_\Pi^{\mathfrak{D}}(I) \subseteq I$

· $T_\Pi^{\mathfrak{D}}$ es continuo y su mínimo punto fijo es el menor modelo libre de Π sobre $\mathfrak{D}$: $I_\Pi^{\mathfrak{D}}$

· Para $\alpha \in B_\Pi^{\mathfrak{D}}$: $\alpha \in I_\Pi^{\mathfrak{D}} \iff \Pi \cup \Delta_{\mathfrak{D}} \models \alpha$

(equivalencia entre las semánticas declarativa y denotacional).

Con estas definiciones podemos hablar del mínimo punto fijo definible (en $\mathcal{M}$) del operador $T_\Pi^{\mathfrak{D}}$ para cualquier estructura. Esta es la semántica alternativa que se va a proponer.

En una estructura standard $\mathfrak{M}_{\mathfrak{D}}$ (cfr. definición 3.3.2) el mínimo punto fijo de $T_{\Pi}^{\mathfrak{D}}$ es definible (y es un modelo libre de Π sobre $\mathfrak{D}$). Esta afirmación será demostrada más adelante.

En general no en todas las estructuras de tipo td $\mathfrak{M} = (\mathcal{T}, \mathfrak{D}, S, ext^{\mathfrak{M}})$ existe el mínimo punto fijo definible en $\mathfrak{M}$ del operador $T_{\Pi}^{\mathfrak{D}}$. Vamos a definir ciertas estructuras (las estructuras admisibles) en las cuales no solo existe el mínimo punto fijo definible, sino que además la fórmula que lo define corresponde con la semántica de Π en $\mathfrak{M}$ según la lógica NPL.

Definición 3.3.9.- AX_t, I_t, AX_s

$$AX_t = \{\ A1:\ \forall i \neg(i + 1 \doteq 0)\ ,$$

$$A2:\ \forall i \forall j\ (i + 1 \doteq j + 1 \longrightarrow i \doteq j)\ ,$$

$$A3:\ \forall i\ (i + 0 \doteq 0)\ ,$$

$$A4:\ \forall i \forall j\ (i + (j + 1) \doteq (i + j) + 1)\ \}$$

AX_t son los axiomas que vamos a exigir a las estructuras temporales $\mathcal{T}$

$I_t = \{\ \underline{ind}_t(\varphi)\ /\ \varphi \in F_{td}\}$ donde

$\underline{ind}_t(\varphi) = (\varphi[0/i] \wedge \forall j\ (\varphi[j/i] \longrightarrow \varphi[j+1/i])) \longrightarrow \forall i\ \varphi$

con j no libre en φ .

$AX_s = \{\ \underline{unit},\ \underline{conc}\}$ donde

$\underline{unit} = \forall x \exists u \forall i\ u(i) \doteq x$

$\underline{conc} = \forall u \forall i \forall v \forall j \exists w \forall k((k \leq i \longrightarrow w(k) \doteq u(k)) \wedge (i < k \leq i+j+1 \longrightarrow \exists l(l \leq j \wedge 1+i+1 \doteq k \wedge w(k) \doteq v(l))))$

unit indica que existen todas las sucesiones constantes y conc que las sucesiones se pueden concatenar.

Son los axiomas que vamos a exigir para S y ext.

$AX_e = \{ \neg \underline{m} \doteq \underline{n} \;/\; m \neq n \;,\; m, n \in N \}$ axiomas de etiquetas.

Definición 3.3.10.- Estructuras admisibles de tipo td

Una estructura admisible de tipo td es cualquier estructura $\mathfrak{M}$ de tipo td con $\mathfrak{M} \models AX_t \cup I_t \cup AX_s \cup AX_e$

En lo que sigue vamos a necesitar una serie de propiedades acerca de 0, +, 1, $\leqslant$, $<$ en estructuras admisibles.

P1: $\forall i(\neg i \doteq 0 \longrightarrow \exists j(j + 1 \doteq i))$

P2: $\forall i \forall j(i + j \doteq j + i)$

P3: $\forall i \forall j \forall k(i + (j + k) \doteq (i + j) + k)$

P4: $\forall i \forall j \forall k\ (i + k \doteq j + k \longrightarrow i \doteq j)$

P5: $\forall i \forall j(i + j \doteq 0 \longrightarrow i \doteq 0 \wedge j \doteq 0)$

P6: $\forall i\ i \leqslant i$

P7: $\forall i \forall j \forall k(i \leqslant j \wedge j \leqslant k \longrightarrow i \leqslant k)$

P8: $\forall i \forall j(i < j \vee i \doteq j \vee j < i)$

P9: $\forall i \forall j(i < j \longleftrightarrow i + 1 \leqslant j)$

P10: $\forall k \forall i(k \leqslant i \vee i + 1 \leqslant k)$

P11: $\forall j \forall k \forall i(k < j \longrightarrow k + i < j + i)$

P12: $\forall i \forall j(i \leqslant j \longleftrightarrow i < j + 1)$

P13: $\forall k \forall j(j \leqslant k + 1 \longleftrightarrow j \doteq k + 1 \vee j \leqslant k)$

P14: $\forall i\ 0 \leqslant i$

P15: $\forall j(j \leqslant 0 \longrightarrow j \doteq 0)$

P16: $\{ \underline{ind}_t'(\varphi) \;/\; \varphi \in F_{td} \}$ con

$\underline{ind}_t'(\varphi) = \forall j(\forall k(k < j \rightarrow \varphi[k/i]) \rightarrow \varphi[j/i]) \rightarrow \forall i \varphi$, j,k no libres en φ

Lema 3.3.1.-

$$AX_t \cup I_t \models \{P1, P2, \ldots\ldots, P16\}$$

Demostración.-

Veremos solo algunas con detalle. El resto se prueban de manera análoga.

P1: Aplicamos $\underline{ind}_t$ a $\varphi = \neg i \doteq 0 \rightarrow \exists j(j + 1 \doteq i)$

$\varphi[0/i] = \neg 0 \doteq 0 \rightarrow \exists j(j + 1 \doteq 0)$

y $AX_t \cup I_t \models \varphi[0/i]$ pues $0 \doteq 0$ es una fórmula válida.

$\varphi[k/i] \rightarrow \varphi[k+1/i]$

$\varphi[k+1/i] = \neg k + 1 \doteq 0 \rightarrow \exists j(j + 1 \doteq k + 1)$ es trivialmente cierta.

P4: Aplicamos $\underline{ind}_t$ a $\varphi = \forall i \forall j(i + k \doteq j + k \rightarrow i \doteq j)$

$\varphi[0/k] = \forall i \forall j(i + 0 \doteq j + 0 \rightarrow i \doteq j)$

y es cierta por A3

$\varphi[k'/k] \rightarrow \varphi[k'+1/k]$

$\varphi[k'+1/k] = \forall i \forall j(i + (k' + 1) \doteq j + (k' + 1) \rightarrow i \doteq j)$

Por A4 $\left.\begin{array}{l} i + (k' + 1) \doteq (i + k') + 1 \\ j + (k' + 1) \doteq (j + k') + 1 \end{array}\right\}$ y por A2

$i + (k' + 1) \doteq j + (k' + 1) \rightarrow i + k' \doteq j + k'$ y por hipótesis de inducción $\varphi[k'/k] \rightarrow \varphi[k'+1/k]$

P16: $\underline{ind}_t'(\varphi) = \forall j(\forall k\ (k < j \rightarrow \varphi[k/i]) \rightarrow \varphi[j/i]) \rightarrow \forall i \varphi$

con j,k no libres en φ

Consideremos la fórmula $\Psi = \forall j(j \leq i \rightarrow \varphi[j/i])$ (donde j no libre en φ)

Probaremos $\underline{ind}_t'(\varphi)$ a partir de $\underline{ind}_t(\Psi)$

Supongamos $\forall j(\forall k\,(k < j \rightarrow \varphi[k/i]) \rightarrow \varphi[j/i])$ (1)

Hemos de deducir $\forall i \varphi$. Para ello es suficiente probar $\forall i \Psi$ puesto que $\Psi \rightarrow \varphi$ gracias a que $i \leq i$ (P6). Usaremos $\underline{ind}_t(\Psi)$

$\Psi[0/i]$

$\Psi[0/i] = \forall j[j \leq 0 \rightarrow \varphi[j/i])$ que equivale a $\varphi[0/i]$ por P15.

Usando A1 tenemos $\forall k \neg k < 0$ y por tanto $\varphi[0/i]$ se concluye de (1)

$\Psi[i'/i] \rightarrow \Psi[i'+1/i]$

$\Psi[i'/i] = \forall j(j \leq i' \rightarrow \varphi[j/i])$

Por P12 esto equivale a $\forall j(j < i'+1 \rightarrow \varphi[j/i])$.

Por (1) obtenemos $\varphi[i'+1/i]$. De esto y $\Psi[i'/i]$ obtenemos con P13 $\Psi[i'+1/i]$

Definición 3.3.11.-

Sea $\mathcal{M} = (\mathcal{T}, \mathcal{D}, S, ext^{\mathcal{M}})$ una estructura de tipo td y Π un programa lógico simple. Diremos que una interpretación libre $I \subseteq B_n^{\mathcal{D}}$ sobre $\mathcal{D}$ es definible en $\mathcal{M}$ si existen fórmulas $\varphi_{X_n^m}(x_1,\ldots,x_m)$ (una por cada variable $X_n^m \in V_\Pi$) tales que para $a_1,\ldots,a_m \in D$ cualesquiera y $X_n^m \in V_\Pi$ se verifica:

$$X_n^m(\underline{a}_1,\ldots,\underline{a}_m) \in I \iff \mathcal{M} \models \varphi_{X_n^m}[\underline{a}_1/x_1,\ldots,\underline{a}_m/x_m]$$

Teorema 3.3.3.-

Si $\mathcal{M} = (\mathcal{T}, \mathcal{D}, S, ext^{\mathcal{M}})$ es una estructura admisible de tipo td y Π es un programa lógico simple, entonces $T_{\Pi}^{\mathcal{D}}$ tiene un mínimo punto fijo $I_{\Pi}^{\mathcal{M}}$ definible en $\mathcal{M}$. Además para cada $X_n^m \in V_n$ puede tomarse $\varphi_{X_n^m}(x_1,\ldots,x_m)$ como la fórmula de F_{td} equivalente a $\Pi X_n^m(x_1,..,x_m)$ es decir:

$$\varphi_{X_n^m}(x_1,\ldots,x_m) = \exists u_1,\ldots,u_r,v,w,i(\underline{demostración}_{\Pi}(u_1,\ldots,u_r,v,w,i) \wedge v(i) \doteq \underline{n} \wedge w(i) \doteq \underline{m} \wedge u_1(i) \doteq x_1 \wedge \ldots \wedge u_m(i) \doteq x_m)$$

siendo r el grado de Π.

Demostración.-

Enunciaremos y demostraremos un lema previo sobre el comportamiento de la fórmula $\underline{demostración}_{\Pi}$ en estructuras admisibles necesario para el teorema.

Dados $u_1,\ldots,u_r,v,w,i$ tales que $\mathcal{M} \models \underline{demostración}_{\Pi}(u_1,..,u_r,v,w,i)$ denotamos por $C^{\mathcal{D}}(u_1,..,u_r,v,w,i)$ al conjunto de átomos $X_n^m(\underline{a}_1,..,\underline{a}_m)$ de $B_{\Pi}^{\mathcal{D}}$ tales que para algún $j \leq^{\mathcal{M}} i$ se tenga $v(j) = \underline{n}$, $w(j) = \underline{m}$, $u_1(j) = \underline{a}_1,\ldots,u_m(j) = \underline{a}_m$.

Nótese que hemos usado $u_1,\ldots,u_r,v,w,i$ para denotar elementos de S y T, cuando son nombres de variables. En lo que sigue, y mientras no haya lugar a error, seguiremos usándolo con el fin de no complicar la notación.

Lema 3.3.2.-

Sea $\mathcal{M} = (\mathcal{T}, \mathcal{D}, S, ext^{\mathcal{M}})$ estructura admisible de tipo td y sea Π un programa lógico simple con grado $(\Pi) = r$.

(1) Sea $u_1,\ldots,u_r,v,w,i$ con $\mathfrak{M} \models \underline{\text{demostración}}_\pi (u_1,\ldots,u_r,v,w,i)$

Entonces para cualquier j con $j \leq^{\mathfrak{M}} i$ $\mathfrak{M} \models \underline{\text{demostración}}_\pi (u_1,..,u_r,v,w,j)$

(2) Sean $u_1',\ldots,u_r',v',w',i,u_1'',\ldots,u_r'',v'',w'',i'$ con

$\mathfrak{M} \models \underline{\text{demostración}}_\pi (u_1',\ldots,u_r',v',w',i)$

$\mathfrak{M} \models \underline{\text{demostración}}_\pi (u_1'',\ldots,u_r'',v'',w'',i')$

Entonces existen $u_1,\ldots,u_r,v,w$ con

$\mathfrak{M} \models \underline{\text{demostración}}_\pi (u_1,\ldots,u_r,v,w,i+i'+1) \wedge \forall j((j \leq i \rightarrow$

$u_1(j) \doteq u_1'(j) \wedge \ldots \wedge u_r(j) \doteq u_r'(j) \wedge v(j) \doteq v'(j) \wedge$

$w(j) \doteq w'(j)) \wedge (i < j \leq i+i'+1 \rightarrow \exists k(k \leq i \wedge k+i'+1 \doteq j \wedge$

$v(j) \doteq v''(k) \wedge w(j) \doteq w''(k) \wedge u_1(j) \doteq u_1''(k) \wedge \ldots \wedge$

$u_r(j) \doteq u_r''(k))))$

(3) Sean $u_1',\ldots,u_r',v',w',i$ con

$\mathfrak{M} \models \underline{\text{demostración}}_\pi (u_1',\ldots,u_r',v',w',i)$

y $X_n^m(\underline{a}_1,\ldots,\underline{a}_m) \in T_\pi^{\mathfrak{D}} (C^{\mathfrak{D}}(u_1',\ldots,u_r',v',w',i))$

Entonces existen $u_1,\ldots,u_r,v,w$ con

$\mathfrak{M} \models \underline{\text{demostración}}_\pi (u_1,\ldots,u_r,v,w,i+1) \wedge u_1(i+1) \doteq \underline{a}_1 \wedge \ldots \wedge$

$u_m(i+1) \doteq \underline{a}_m \wedge v(i+1) \doteq \underline{n} \wedge w(i+1) \doteq \underline{m} \wedge \forall j(j \leq i \rightarrow$

$u_1(j) \doteq u_1'(j) \wedge \ldots \wedge u_r(j) \doteq u_r'(j) \wedge v(j) \doteq v'(j) \wedge$

$w(j) \doteq w'(j))$

Demostración.-

(1) Es inmediato a partir de la definición y del lema 3.3.1

(2) Como $\mathfrak{M}$ admisible, $\mathfrak{M} \models \underline{\text{conc}}$ y por tanto existen $u_1,\ldots,u_r$,

v,w con

$$\mathfrak{M} \models \forall j((j \leq i \rightarrow u_1(j) \doteq u_1'(j) \wedge \ldots \wedge u_r(j) \doteq u_r'(j) \wedge v(j) \doteq v'(j) \wedge w(j) \doteq w'(j)) \wedge (i < j \leq i+i'+1 \rightarrow \exists k(k \leq i \wedge k+i'+1 \doteq j \wedge u_1(j) \doteq u_1''(k) \wedge \ldots \wedge u_r(j) \doteq u_r''(k) \wedge v(j) \doteq v''(k) \wedge w(j) \doteq w''(k)))) \qquad (1)$$

Además

$$\mathfrak{M} \models \underline{\text{demostración}}_{\pi}(u_1',\ldots,u_r',v',w',i) \wedge \underline{\text{demostración}}_{\pi}(u_1'',\ldots,u_r'',v'',w'',i')$$

Luego

$$\mathfrak{M} \models \forall j((j \leq i \rightarrow \bigvee_{C \in \Pi} \gamma_C(u_1',..,u_r',v',w',j)) \wedge (j \leq i' \rightarrow \bigvee_{C \in \Pi} \gamma_C(u_1'',..,u_r'',v'',w'',j)))$$

Y por (1) y el lema 3.3.1

$$\mathfrak{M} \models \forall j((j \leq i \rightarrow \bigvee_{C \in \Pi} \gamma_C(u_1,..,u_r,v,w,j)) \wedge (i < j \leq i+i'+1 \rightarrow \bigvee_{C \in \Pi} \gamma_C(u_1,..,u_r,v,w,j)))$$

Y de nuevo con las propiedades del lema 3.3.1

$$\mathfrak{M} \models \forall j(j \leq i+i'+1 \rightarrow \bigvee_{C \in \Pi} \gamma_C(u_1,\ldots,u_r,v,w,j))$$

y por tanto

$$\mathfrak{M} \models \underline{\text{demostración}}_{\pi}(u_1,\ldots,u_r,v,w,i+i'+1)$$

(3) Como $\mathfrak{M}$ admisible $\mathfrak{M} \models \underline{\text{unit}}$, luego existen $u_1'',\ldots,u_r'',v'',w''$ con $\mathfrak{M} \models \forall j(u_1''(j) \doteq \underline{a}_1 \wedge .. \wedge u_m''(j) \doteq \underline{a}_m \wedge v''(j) \doteq \underline{n} \wedge w''(j) \doteq \underline{m})$

Si ahora aplicamos $\underline{\text{conc}}$ a $u_1',\ldots,u_r',v',w',u_1'',\ldots,u_r'',v'',w''$ tenemos que existen $u_1,\ldots,u_r,v,w$ con

$$\mathfrak{M} \models \forall j(j \leq i \rightarrow u_1(j) \doteq u_1'(j) \wedge \ldots \wedge u_r(j) \doteq u_r'(j) \wedge v(j) \doteq v'(j)$$

Llamamos $I_\Pi^{\mathfrak{M}}$ a la interpretación libre de Π sobre $\mathfrak{D}$ definida en $\mathfrak{M}$ por las fórmulas $\Pi X_n^m(x_1,\ldots,x_m)$ con $X_n^m \in V_\Pi$

Para obtener el teorema basta probar:

(1) $I_\Pi^{\mathfrak{M}} \subseteq T_\Pi^{\mathfrak{D}}(I_\Pi^{\mathfrak{M}})$

(2) $T_\Pi^{\mathfrak{D}}(I_\Pi^{\mathfrak{M}}) \subseteq I_\Pi^{\mathfrak{M}}$

$\Big\}$ $I_\Pi^{\mathfrak{M}}$ punto fijo de $T_\Pi^{\mathfrak{D}}$

(3) $I_\Pi^{\mathfrak{M}} \subseteq I$ para cualquier I punto fijo de $T_\Pi^{\mathfrak{D}}$ definible en $\mathfrak{M}$

(1) $I_\Pi^{\mathfrak{M}} \subseteq T_\Pi^{\mathfrak{D}}(I_\Pi^{\mathfrak{M}})$

Sea $X_n^n(\underline{a}_1,\ldots,\underline{a}_m) \in I_\Pi^{\mathfrak{M}}$

$$\mathfrak{M} \models \exists u_1,\ldots,u_r,v,w,i(\underline{\text{demostración}}_\Pi(u_1,\ldots,u_r,v,w,i) \wedge v(i) \doteq \underline{n} \wedge w(i) \doteq \underline{m} \wedge u_1(i) \doteq \underline{a}_1 \wedge \ldots \wedge u_m(i) \doteq \underline{a}_m)$$

Por tanto para $u_1,\ldots,u_r,v,w,i$

$$\mathfrak{M} \models \forall j(j \leq i \rightarrow \bigvee_{C\in\Pi} \gamma_C(u_1,\ldots,u_r,v,w,j))$$

En particular para $j = i$ usando el lema 3.3.1 existe $C \in \Pi$ con

$$\mathfrak{M} \models \gamma_C(u_1,\ldots,u_r,v,w,i) \wedge u_1(i) \doteq \underline{a}_1 \wedge \ldots \wedge u_m(i) \doteq \underline{a}_m \wedge v(i) \doteq \underline{n} \wedge w(i) \doteq \underline{m}$$

Es decir existe una instancia básica de C

$X_n^m(\underline{a}_1,\ldots,\underline{a}_m) \leftarrow \beta_1,\ldots,\beta_q$ con

β_j primitivo $\Rightarrow \mathfrak{M} \models \beta_j \Rightarrow \beta_j \in \Delta_{\mathfrak{D}}$

$\beta_j = X_{n'}^{m'}(\underline{b}_1,\ldots,\underline{b}_{m'}) \Rightarrow \mathfrak{M} \models \exists j'(j' < i \wedge u_1(j') \doteq \underline{b}_1 \wedge \ldots \wedge u_{m'}(j') \doteq \underline{b}_{m'} \wedge v(j') \doteq \underline{n}' \wedge w(j') \doteq \underline{m}')$

Ahora por el lema 3.3.2 (1) se tiene

$$\mathfrak{M} \models \underline{\text{demostración}}_\Pi(u_1,..,u_r,v,w,j') \wedge u_1(j') \doteq \underline{b}_1 \wedge \ldots \wedge u_{m'}(j') \doteq \underline{b}_{m'} \wedge v(j') \doteq \underline{n}' \wedge w(j') \doteq \underline{m}'$$

$\wedge\ w(j) \doteq w'(j)) \wedge u_1(i+1) \doteq \underline{a}_1 \wedge \ldots \wedge u_m(i+1) \doteq \underline{a}_m \wedge$

$v(i+1) \doteq \underline{n} \wedge w(i+1) \doteq \underline{m}$ (2)

Además como $X_n^m(\underline{a}_1,\ldots,\underline{a}_m) \in T_\pi^{\mathfrak{D}}$ $(C^{\mathfrak{D}}(u_1',\ldots,u_r',v',w',i)$ existe $X_n^m(\underline{a}_1,\ldots,\underline{a}_m) \leftarrow \beta_1,\ldots,\beta_q$ instancia básica de C cláusula definida de Π con $\{\beta_1,\ldots,\beta_q\} \subseteq \Delta_{\mathfrak{D}} \cup C^{\mathfrak{D}}(u_1',\ldots,u_r',v',w',i)$

Esto supone que para todo j, $1 \leq j \leq q$

β_j primitivo $\Rightarrow \beta_j \in \Delta_{\mathfrak{D}} \Rightarrow \mathcal{M} \models \beta_j$

$\beta_j = X_{n'}^{m'}(\underline{b}_1,\ldots,\underline{b}_m) \Rightarrow X_{n'}^{m'}(\underline{b}_1,\ldots,\underline{b}_m) \in C^{\mathfrak{D}}(u_1',\ldots,u_r',v',w',i)$

$\Rightarrow$ existe j $j \leq^{\mathcal{M}} i$ y $v'(j) = \underline{n}', w'(j) = m'$ y $u_1'(j) = \underline{b}_1,\ldots,$

$u_{m'}'(j) = \underline{b}_{m'} \Rightarrow$ (De (2) y las propiedades del lema 3.3.1)

$\mathcal{M} \models \exists j(j < i+1 \wedge v(j) \doteq \underline{n}' \wedge w(j) \doteq \underline{m}' \wedge u_1(j) \doteq \underline{b}_1 \wedge \ldots \wedge$

$u_{m'}(j) \doteq \underline{b}_{m'})$

Luego $\mathcal{M} \models \gamma_C(u_1,\ldots,u_r,v,w,i+1)$

Como $\mathcal{M} \models \underline{\text{demostración}}_\pi(u_1',\ldots,u_r',v',w',i)$ tenemos

$\mathcal{M} \models \forall j(j \leq i \rightarrow \bigvee_{C \in \pi} \gamma_C(u_1',\ldots,u_r',v',w',j))$ y por (2)

$\mathcal{M} \models \forall j(j \leq i \rightarrow \bigvee_{C \in \pi} \gamma_C(u_1,\ldots,u_r,v,w,j))$ y con el lema 3.3.1 concluimos $\mathcal{M} \models \forall j(j \leq i+1 \rightarrow \bigvee_{C \in \pi} \gamma_C(u_1,\ldots,u_r,v,w,j))$ luego

$\mathcal{M} \models \underline{\text{demostración}}_\pi(u_1,\ldots,u_r,v,w,i+1)$

El lema afirma:

(1) Todo prefijo de una "demostración" es una"demostración"

(2) Las "demostraciones" se pueden concatenar.

(3) Una "demostración" se puede ampliar aplicando $T_\pi^{\mathfrak{D}}$ a lo ya obtenido

y por tanto $X_n^{m'}(\underline{b}_1,\ldots,\underline{b}_m) \in I_\Pi^{\mathcal{M}}$

Con lo cual $X_n^m(\underline{a}_1,\ldots,\underline{a}_m) \in T_\Pi^{\mathfrak{D}}(I_\Pi^{\mathcal{M}})$

(2) $T_\Pi^{\mathfrak{D}}(I_\Pi^{\mathcal{M}}) \subseteq I_\Pi^{\mathcal{M}}$

Sea $X_n^m(\underline{a}_1,\ldots,\underline{a}_m) \in T_\Pi^{\mathfrak{D}}(I_\Pi^{\mathcal{M}}) \Longrightarrow$ existe una instancia básica de una claúsula definida de Π $X_n^m(\underline{a}_1,\ldots,\underline{a}_m) \longleftarrow \beta_1,\ldots,\beta_q$ con $\{\beta_1,\ldots,\beta_q\} \subseteq \Delta\mathfrak{D} \cup I_\Pi^{\mathcal{M}}$

Esto supone que para cada $\beta \in \{\beta_1,\ldots,\beta_q\}$

β primitivo $\Longrightarrow \beta \in \Delta\mathfrak{D}$

β no primitivo $\Longrightarrow \beta = X_{n_\beta}^{m_\beta}(\underline{b}_1^\beta,\ldots,\underline{b}_m^\beta) \in I_\Pi^{\mathcal{M}} \Longleftrightarrow$

$\mathcal{M} \models \exists u_1^\beta,\ldots,u_r^\beta,v^\beta,w^\beta,i^\beta$ ($\underline{\text{demostración}}_\Pi(u_1^\beta,..,u_r^\beta,v^\beta,w^\beta,i^\beta)$

$\wedge\ v^\beta(i^\beta) \doteq \underline{n}^\beta \wedge w^\beta(i^\beta) \doteq \underline{m}^\beta \wedge u_1^\beta(i^\beta) \doteq \underline{b}_1^\beta \wedge \ldots \wedge$

$u_{m_\beta}^\beta(i^\beta) \doteq \underline{b}_{m_\beta}$)

Aplicamos sucesivamente el lema 3.3.2 (2) para los β con $\beta \in \{\beta_1,\ldots,\beta_q\}$ átomos no primitivos y existen $u_1',\ldots,u_r',v',w',i'$ con $\mathcal{M} \models \underline{\text{demostración}}_\Pi(u_1',\ldots,u_r',v',w',i')$ y para cada β con β átomo no primitivo $X_{n\beta}^{m\beta}(\underline{b}_1^\beta,\ldots,\underline{b}_{m\beta}^\beta) \in C^{\mathfrak{D}}(u_1',..,u_r',v',w',i')$

Por tanto $X_n^m(\underline{a}_1,\ldots,\underline{a}_m) \in T_\Pi^{\mathfrak{D}}(C^{\mathfrak{D}}(u_1',\ldots,u_r',v',w',i'))$

y por el lema 3.3.2 (3) existen $u_1,\ldots,u_r,v,w,i$ con

$\mathcal{M} \models \underline{\text{demostración}}_\Pi(u_1,\ldots,u_r,v,w,i) \wedge u_1(i) \doteq \underline{a}_1 \wedge \ldots \wedge u_m(i) \doteq \underline{a}_m$

$\wedge\ v(i) \doteq \underline{n} \wedge w(i) \doteq \underline{m}$

y concluimos $X_n^m(\underline{a}_1,\ldots,\underline{a}_m) \in I_\Pi^{\mathcal{M}}$

(3) $I_\Pi^{\mathcal{M}} \subseteq I$ para cualquier I punto fijo definible en $\mathcal{M}$ de $T_\Pi^{\mathfrak{D}}$

Como I es definible existen fórmulas $\varphi'_{X_n^m}(x_1,\ldots,x_m) \in F_{td}$ para

cada $X_n^m \in V_\pi$ con

$X_n^m(\underline{a}_1,\ldots,\underline{a}_m) \in I \iff \mathcal{M} \models \varphi'_{X_n^m}[\underline{a}_1/x_1,\ldots,\underline{a}_m/x_m]$

$\varphi_{X_n^m}(x_1,\ldots,x_m)$ es de la forma

$\exists i\, \exists u_1,\ldots,u_r,v,w(\underline{\text{demostración}}_\pi(u_1,\ldots,u_r,v,w,i) \wedge v(i) \doteq \underline{n} \wedge$

$w(i) \doteq \underline{m} \wedge u_1(i) \doteq x_1 \wedge \ldots \wedge u_r(i) \doteq x_m)$

que abreviaremos como $\exists i\, \varphi''_{X_n^m}(x_1,\ldots,x_m,i)$

Para probar (3) aplicaremos el axioma de $\underline{\text{ind}}_t'$ (ver lema 3.3.1) a las fórmulas $\varphi''_{X_n^m}(x_1,\ldots,x_m,i) \longrightarrow \varphi'_{X_n^m}(x_1,\ldots,x_m)$

Supongamos $\varphi''_{X_n^m}(x_1,\ldots,x_m,k) \longrightarrow \varphi'_{X_n^m}(x_1,\ldots,x_m)$ para cualquier $k < j$ y probémoslo para j

Si $\mathcal{M} \models \exists u_1,\ldots,u_r,v,w(\underline{\text{demostración}}_\pi(u_1,\ldots,u_r,v,w,j) \wedge v(j) \doteq \underline{n}$

$\wedge\ w(j) \doteq \underline{m} \wedge u_1(j) \doteq \underline{a}_1 \wedge \ldots \wedge u_m(j) \doteq \underline{a}_m)$

como $\underline{\text{demostración}}_\pi(u_1,..,u_r,v,w,j) = \forall j'(j' \leq j \longrightarrow \bigvee_{C\in\Pi} \gamma_C(u_1,\ldots,u_r,v,w,j'))$

en particular para j (lema 3.3.1) obtenemos

$\mathcal{M} \models \bigvee_{C\in\Pi} \gamma_C(u_1,\ldots,u_r,v,w,j)$, luego existe una instancia básica de una claúsula definida de Π $\quad X_n^m(\underline{a}_1,\ldots,\underline{a}_m) \leftarrow \beta_1,\ldots,\beta_q$ tal que para cada $\beta \in \{\beta_1,\ldots,\beta_q\}$

β primitivo $\Longrightarrow \mathcal{M} \models \beta \Longrightarrow \beta \in \Delta_{\mathfrak{D}}$

$\beta = X_n^{m'}(\underline{b}_1,\ldots,\underline{b}_{m'}) \Longrightarrow \mathcal{M} \models \exists k(k < j \wedge v(k) \doteq \underline{n}' \wedge w(k) \doteq \underline{m}'$

$\wedge\ u_1(k) \doteq \underline{b}_1 \wedge \ldots \wedge u_{m'}(k) \doteq \underline{b}_{m'})$

y por el lema 3.3.2 (1) tenemos

$\mathcal{M} \models \exists u_1,\ldots,u_r,v,w(\underline{\text{demostración}}_{\Pi}(u_1,\ldots,u_r,v,w,k) \wedge v(k) \doteq \underline{n}'$

$\wedge\ w(k) \doteq \underline{m}' \wedge u_1(k) \doteq \underline{b}_1 \wedge \ldots \wedge u_m(k) \doteq \underline{b}_m)$

Es decir $\mathcal{M} \models \varphi''_{X^m_{n'}}[\underline{b}_1/x_1,\ldots,\underline{b}_m/x_m,k/i]$ y por hipótesis de inducción

$\mathcal{M} \models \varphi'_{X^m_{n'}}[\underline{b}_1/x_1,\ldots,\underline{b}_m/x_m]$ es decir $\beta \in I$

y esto para cada β no primitivo.

Luego $X^m_n(\underline{a}_1,\ldots,\underline{a}_m) \in T^{\mathfrak{D}}_{\Pi}(I)$ y por ser I punto fijo $T^{\mathfrak{D}}_{\Pi}(I) \subseteq I$

y por tanto $\mathcal{M} \models \varphi'_{X^m_n}[\underline{a}_1/x_1,\ldots,\underline{a}_m/x_m]$

Notese que también hemos probado que $I^{\mathcal{M}}_{\Pi}$ es menor que todas las $I \in B^{\mathfrak{D}}_{\Pi}$ definibles en $\mathcal{M}$ que verifiquen $T^{\mathfrak{D}}_{\Pi}(I) \subseteq I$.

Corolario 3.3.1.-

Si $\mathcal{M}_{\mathfrak{D}}$ es la estructura standard sobre $\mathfrak{D}$ (estructura arbitraria de tipo d) entonces $I^{\mathfrak{D}}_{\Pi}$ es definible en $\mathcal{M}_{\mathfrak{D}}$ y corresponde a la semántica de Π en $\mathcal{M}_{\mathfrak{D}}$ según NPL.

Demostración.-

$\mathcal{M}_{\mathfrak{D}}$ es una estructura admisible y según el teorema la semántica de Π en $\mathcal{M}_{\mathfrak{D}}$ corresponde al menor punto fijo $I^{\mathcal{M}_{\mathfrak{D}}}_{\Pi}$ de $T^{\mathfrak{D}}_{\Pi}$ definible en $\mathcal{M}_{\mathfrak{D}}$. Por otra parte, la definición de $\mathcal{M}_{\mathfrak{D}}$ y la caracterización del menor punto fijo $I^{\mathfrak{D}}_{\Pi}$ de $T^{\mathfrak{D}}_{\Pi}$ como $T^{\mathfrak{D}}_{\Pi}\uparrow_{\omega}$ garantizan que $I^{\mathfrak{D}}_{\Pi}$ corresponde a la semántica de Π en $\mathcal{M}_{\mathfrak{D}}$.

Luego $I^{\mathcal{M}_{\mathfrak{D}}}_{\Pi} = I^{\mathfrak{D}}_{\Pi} = T^{\mathfrak{D}}_{\Pi}\uparrow_{\omega}$ es definible en $\mathcal{M}_{\mathfrak{D}}$ según la semántica NPL de Π en $\mathcal{M}_{\mathfrak{D}}$.

El resultado obtenido asi afirma que para cualquier estructura admisible nuestra semántica NPL de programas lógicos define el mínimo punto fijo definible de cierto operador asociado al programa.

Todo esto es análogo a lo desarrollado por Cartwright (ver la sección 2.5) para sus programas recursivos, pero, como veremos en 3.4, nuestros programas son, en cierto sentido, más generales que aquéllos. Además evitamos algunos de sus problemas. En la sección 2.5 definíamos los programas recursivos

$F_1 = \{ g(x) = g(x) \}$ y $F_2 = \{ g(x) = g(x) + 1 \}$

para el lenguaje de la aritmética. Ambos definen g como la función totalmente indefinida, por lo que se tiene $\forall x \forall y \neg G(x,y)$. Pero usando una axiomatización $AX_{\mathfrak{N}}$ para $\mathfrak{N}$ (modelo standard de la aritmética) axiomas de inducción y la ecuación de F_1 no podiamos deducirlo, pues $T^{\mathfrak{D}}_{F_1}$ tiene otros puntos fijos. Para F_2 si conseguiamos deducirlo pues $T^{\mathfrak{D}}_{F_2}$ tiene un único punto fijo.

En nuestro planteamiento, F_1 y F_2 corresponden a los dos programas lógicos:

$\Pi_1 = \{ X(x,y) \leftarrow X(x,y) \}$ $\Pi_2 = \{ X(x,y+1) \leftarrow X(x,y) \}$

y usando inducción sobre la longitud de las "demostraciones" podemos demostrar que :

$\forall x \forall y \neg \Pi_1 X(x,y)$ y que $\forall x \forall y \neg \Pi_2 X(x,y)$.

Como conclusión de lo trabajado en esta sección, tenemos que nuestra semántica NPL para programas lógicos se comporta como la semántica natural en estructuras standard (que generaliza la semántica de programas de Horn sobre interpretaciones de Herbrand). Incluso es válida para trabajar en estructuras no standard siempre que sean admisibles.

3.4.- Potencia de NPL. Aplicación práctica de NPL a la derivación formal de propiedades de programas lógicos.

En esta sección estudiaremos la potencia de NPL y veremos alguna aplicación de LNP para demostrar ciertas propiedades de programas lógicos.

Para comprobar la potencia de NPL simularemos los programas regulares de la definición 1.2.1 y los programas recursivos de la definición 2.5.2 como programas lógicos. Probaremos que en estructuras standard y en estructuras admisibles esta simulación es adecuada.

En la práctica interesa conocer el comportamiento de los programas en estructuras standard sobre tipos de datos accesibles. Puesto que tales estructuras son admisibles, todo lo que se pruebe en LNP apoyándose en los axiomas de admisibilidad sera válido en el caso standard. Además necesitaremos apoyarnos en algunas otras propiedades de la estructura de datos standard que nos interesen en cada caso (si $d = \{0, suc, +, \cdot, \leq\}$ podemos considerar los axiomas de Peano ; si $\mathfrak{D}$ es finitamente generable con un conjunto de funciones $G \subseteq d$, G finito, podemos pedir el esquema de induccion:

$$\underline{ind}_d(\varphi) = \bigwedge_{g \in G} \forall z_1, \ldots, z_{\#g} \left(\bigwedge_{i=1}^{\#g} \varphi[z_i/x] \rightarrow \varphi[g(z_1, \ldots, z_{\#g})/x] \right) \rightarrow \forall x \varphi$$

donde $z_1, \ldots, z_{\#g}$ no aparecen libres en φ).

Todas estas propiedades las expresaremos con un conjunto de axiomas AX y trabajaremos con $AX \vdash^{LNP}$ similarmente a lo que hacíamos con NDL en la sección 2.4

Simulación de programas regulares como programas lógicos.

Los programas regulares se interpretan como relaciones entre estados (cambios en las variables del programa),

Para simular un programa α definiremos un programa lógico Π_α, con una única variable de programa en $V_{\Pi\alpha}$, X_α^{2m} si $\bar{x} = (x_1,\ldots,x_m) = var(\alpha)$ son las variables del programa α.

Intuitivamente X_α^{2m} representa el grafo de α en el sentido de que $\quad \bar{x}\,\alpha\,\bar{y} \iff X_\alpha^{2m}(\bar{x},\bar{y})$

Definiremos Π_α recursivamente sobre la estructura de α,

Podemos suponer que las variables de todos los subprogramas de son también $\bar{x}$.

$\alpha = (x_i := \tau)$

$$\Pi_\alpha = \left\{ X_\alpha^{2m}(\bar{x},x_1,\ldots,x_{i-1},\tau,x_{i+1},\ldots,x_m) \leftarrow \right\}$$

$\alpha = \varphi?$ procedemos por inducción sobre la estructura de φ.

Si φ es de primer orden libre de cuantificadores podemos poner φ en forma normal disyuntiva $\varphi \leftrightarrow \varphi_1 \vee \ldots \vee \varphi_n$ y cada φ_i es una conjunción de átomos primitivos $\varphi_i = \beta_1^i \wedge \ldots \wedge \beta_{q_i}^i$

$$\Pi_{\varphi?} = \left\{ \begin{array}{ll} X_{\varphi?}^{2m}(\bar{x},\bar{x}) \leftarrow \beta_1^1,\ldots,\beta_{q_1}^1 & , \\ \vdots & \vdots \\ X_{\varphi?}^{2m}(\bar{x},\bar{x}) \leftarrow \beta_1^i,\ldots,\beta_{q_i}^i & , \\ \vdots & \vdots \\ X_{\varphi?}^{2m}(\bar{x},\bar{x}) \leftarrow \beta_1^n,\ldots,\beta_{q_n}^n & \end{array} \right\}$$

$\varphi = \psi \wedge \chi$

$\Pi_{\varphi?} = \{ X^{2m}_{\varphi?}(\bar{x},\bar{x}) \leftarrow \Pi_{\psi?} X^{2m}_{\psi?}(\bar{x},\bar{x}) \ , \Pi_{\chi?} X^{2m}_{\chi?}(\bar{x},\bar{x}) \}$

$\varphi = \neg \psi$

$\Pi_{\varphi?} = \{ X^{2m}_{\varphi?}(\bar{x},\bar{x}) \leftarrow \neg\Pi_{\psi?} X^{2m}_{\psi?}(\bar{x},\bar{x}) \}$

$\varphi = \exists x_i \psi$

$\Pi_{\varphi?} = \{ X^{2m}_{\varphi?}(\bar{x},\bar{x}) \leftarrow \Pi_{\psi?} X^{2m}_{\psi?}(x_1,..,x_{i-1},y,x_{i+1},..,x_m,x_1,..,x_{i-1},y,..,x_m) \}$

$\varphi = \langle\beta\rangle\psi$

$\Pi_{\varphi?} = \{ X^{2m}_{\varphi?}(\bar{x},\bar{x}) \leftarrow \Pi_{\beta} X^{2m}_{\beta}(\bar{x},\bar{y}) \ , \Pi_{\psi?} X^{2m}_{\psi?}(\bar{y},\bar{y}) \}$

$\alpha = \beta \cup \beta'$

$\Pi_{\alpha} = \{ X^{2m}_{\alpha}(\bar{x},\bar{y}) \leftarrow \Pi_{\beta} X^{2m}_{\beta}(\bar{x},\bar{y}) \ ,$

$X^{2m}_{\alpha}(\bar{x},\bar{y}) \leftarrow \Pi_{\beta'} X^{2m}_{\beta'}(\bar{x},\bar{y}) \}$

$\alpha = \beta ; \beta'$

$\Pi_{\alpha} = \{ X^{2m}_{\alpha}(\bar{x},\bar{y}) \leftarrow \Pi_{\beta} X^{2m}_{\beta}(\bar{x},\bar{z}) \ , \Pi_{\beta} X^{2m}_{\beta}(\bar{z},\bar{y}) \}$

$\alpha = \beta^*$

$\Pi_{\alpha} = \{ X^{2m}_{\alpha}(\bar{x},\bar{x}) \leftarrow \ ,$

$X^{2m}_{\alpha}(\bar{x},\bar{y}) \leftarrow X^{2m}_{\alpha}(\bar{x},\bar{z}) \ , \Pi_{\beta} X^{2m}_{\beta}(\bar{z},\bar{y}) \}$

Si α es un programa regular simple (en el sentido de que φ es de primer orden sin cuantificadores si $\varphi?$ aparece en α) es fácil modificar esta construcción para conseguir que el programa lógico que simula α sea simple.

Un ejemplo de la simulacion es:

$= ((\neg x_1 \doteq x_2)? \ ; \ x_1 := f(x_1))^* \ ; \ (x_1 \doteq x_2)?$

simula con programas regulares el programa

<u>While</u> $(x_1 \neq x_2)$ <u>do</u> $x_1 := f(x_1)$ <u>od</u>

Tomando

$\beta_1 = (\neg x_1 \doteq x_2)?$

$\beta_2 = x_1 := f(x_1)$

$\beta_3 = (x_1 \doteq x_2)?$

$\beta_4 = \beta_1 ; \beta_2$

$\beta_5 = \beta_4^*$

resulta:

$$\Pi_\alpha = \{ X_\alpha(x_1,x_2,y_1,y_2) \leftarrow \Pi_{\beta_5} X_{\beta_5}(x_1,x_2,z_1,z_2), \Pi_{\beta_3} X_{\beta_3}(z_1,z_2,y_1,y_2) \}$$

$$\Pi_{\beta_5} = \{ X_{\beta_5}(x_1,x_2,x_1,x_2) \leftarrow \ , \ X_{\beta_5}(x_1,x_2,y_1,y_2) \leftarrow X_{\beta_5}(x_1,x_2,z_1,z_2), \Pi_{\beta_4} X_{\beta_4}(z_1,z_2,y_1,y_2) \}$$

$$\Pi_{\beta_4} = \{ X_{\beta_4}(x_1,x_2,y_1,y_2) \leftarrow \Pi_{\beta_1} X_{\beta_1}(x_1,x_2,z_1,z_2), \Pi_{\beta_2} X_{\beta_2}(z_1,z_2,y_1,y_2) \}$$

$$\Pi_{\beta_3} = \{ X_{\beta_3}(x_1,x_2,x_1,x_2) \leftarrow x_1 \doteq x_2 \}$$

$$\Pi_{\beta_2} = \{ X_{\beta_2}(x_1,x_2,f(x_1),x_2) \leftarrow \ \}$$

$$\Pi_{\beta_1} = \{ X_{\beta_1}(x_1,x_2,x_1,x_2) \leftarrow \neg x_1 \doteq x_2 \}$$

Teorema 3.4.1.-

Sea $\mathfrak{D}$ estructura de tipo de similaridad d y $\bar{a},\bar{b}$ m-tuplas de elementos de $\mathfrak{D}$, $\alpha \in PS_d$.

$$\bar{a}\ \alpha^{\mathfrak{D}}\ \bar{b} \iff \mathfrak{M}_{\mathfrak{D}} \models \Pi_\alpha X_\alpha^{2m}(\bar{a},\bar{b})$$

(donde $\mathfrak{M}_{\mathfrak{D}}$ es la estructura standard de tipo td sobre $\mathfrak{D}$)

Este teorema demuestra que la simulación de programas regulares como programas lógicos es válida sobre estructuras standard. La prueba se obtiene por inducción sobre la estructura de α .

Además si consideramos una estructura $\mathfrak{M}$ de tipo td e interpretamos $\alpha^{\mathfrak{M}}$ como en la definición 2.2.4 pero sustituyendo los símbolos suc(x) por x + 1 y $\leq$ por su definición en $\mathfrak{M}$ obtenemos el siguiente teorema:

Teorema 3.4.2.-

Si $\mathfrak{M} = (\mathcal{T} , \mathfrak{D} , S, ext^{\mathfrak{M}})$ estructura admisible de tipo td, $\alpha \in PS_d$ y $\bar{a},\bar{b}$ m-tuplas de elementos de $\mathfrak{D}$ se tiene:

$$\bar{a}\ \alpha^{\mathfrak{M}}\ \bar{b} \iff \mathfrak{M} \models \Pi_\alpha X_\alpha^{2m}(\bar{a},\bar{b})$$

Este teorema nos afirma la validez de la simulación también para la semántica NDL de programas regulares. Las hipótesis de admisibilidad de las "trazas internas" para α dan "demostraciones" para Π y viceversa.

Simulación de programas recursivos funcionales como programas lógicos simples.

Sea d un tipo de similaridad y F un programa recursivo de de tipo d $F = \{ f_1(\bar{x}_1) = \tau_1,\ f_2(\bar{x}_2) = \tau_2, \ldots,\ f_n(\bar{x}_n) = \tau_n \}$

con $f_1, \ldots, f_n$ nuevos símbolos de función que no estan en d con f_i de m_i argumentos.

La simulación de un programa recursivo se hará calculando los grafos de cada f_i en las variables de programas $X_{f_i}^{m_i+1}$ que simplificaremos por X_{f_i}. La idea es $f_i(\bar{a}) = b$ si y solo si $X_{f_i}(\bar{a},b)$

Por cada subtérmino $\tau \in TR_d$ de τ_i definiremos un programa Π_τ con una variable de programa $X_\tau^{m_i+1}$ (abreviado X_τ) que lo calcula para los valores de las variables $\bar{x}_i$ $\quad X_\tau(\bar{x}_i,\sigma)$ si $\tau(\bar{x}_i) = \sigma$

Definimos Π_τ recursivamente:

$\tau = x_i^j$

$\Pi_\tau = \{ X_\tau(\bar{x}_i, x_i^j) \leftarrow \}$

$\tau = g(\sigma_1, \ldots, \sigma_m)$

$\Pi_\tau = \{ X_\tau(\bar{x}_i, g(y_1,\ldots,y_m)) \leftarrow X_{\sigma_1}(\bar{x}_i,y_1), \ldots\ldots, X_{\sigma_m}(\bar{x}_i,y_m) \}$

$\cup\ \Pi_{\sigma_1} \cup \ldots\ldots \cup \Pi_{\sigma_m}$

$\tau = f_j(\sigma_1, \ldots, \sigma_{m_j})$

$\Pi_\tau = \{ X_\tau(\bar{x}_i,y) \leftarrow X_{\sigma_1}(\bar{x}_i,y_1), \ldots, X_{\sigma_{m_j}}(\bar{x}_i,y_{m_j}),\ X_{f_j}(y_1,\ldots,y_{m_j},y) \}$

$\cup\ \Pi_{\sigma_1} \cup \ldots\ldots \cup \Pi_{\sigma_m}$

τ = **if** $Q(\sigma_1,\ldots,\sigma_m)$ **then** τ_1' **else** τ_2'

$$\Pi_\tau = \{\ X_\tau(\bar{x}_i,y) \leftarrow X_{\sigma_1}(\bar{x}_i,y_1),\ldots, X_{\sigma_m}(\bar{x}_i,y_m), Q(y_1,\ldots,y_m), X_{\tau_1'}(\bar{x}_i,y)\ ,$$

$$X_\tau(\bar{x}_i,y) \leftarrow X_{\sigma_1}(\bar{x}_i,y_1),\ldots, X_{\sigma_m}(\bar{x}_i,y_m), \neg Q(y_1,\ldots,y_m), X_{\tau_2'}(\bar{x}_i,y)\ \}$$

$$\cup\ \Pi_{\sigma_1} \cup \ldots\ldots \cup \Pi_{\sigma_m} \cup \Pi_{\tau_1'} \cup \Pi_{\tau_2'}$$

El programa lógico Π_F asociado al programa recursivo F es:

$$\Pi_F = \{\ X_{f_1}(\bar{x}_1,y_1) \leftarrow X_{\tau_1}(\bar{x}_1,y_1),$$

$$\vdots \qquad\qquad \vdots$$

$$X_{f_n}(\bar{x}_n,y_n) \leftarrow X_{\tau_n}(\bar{x}_n,y_n)\ \} \cup \Pi_{\tau_1} \cup \ldots \cup \Pi_{\tau_n}$$

Un ejemplo posible es un programa recursivo que calcula el factorial es:

$$F = \{\ f(x) = \text{if } x \doteq 0 \text{ then } 1 \text{ else } x\cdot f(x-1)\ \}$$

y su traducción es:

$$\Pi_F = \{\ X_f(x,y) \leftarrow X_\tau(x,y)\ ,$$

$$X_\tau(x,y) \leftarrow x \doteq 0\ ,\ X_1(x,y)\ ,$$

$$X_\tau(x,y) \leftarrow \neg\, x \doteq 0\ ,\ X_{x\cdot f(x-1)}(x,y)\ ,$$

$$X_{x\cdot f(x-1)}(x,y_1{:}y_2) \leftarrow X_x(x,y_1)\ ,\ X_{f(x-1)}(x,y_2)\ ,$$

$$X_x(x,x) \leftarrow\ ,$$

$$X_{f(x-1)}(x,y) \leftarrow X_f(y_1,y)\ ,\ X_{x-1}(x,y_1)\ ,$$

$$X_{x-1}(x,\ y_1{-}y_2) \leftarrow X_1(x,y_2)\ ,\ X_x(x,y_1)$$

$$X_1(x,1) \leftarrow\ \}$$

Como en el caso de los programas regulares se pueden enunciar teoremas que expliquen en que sentido el programa lógico Π_F representa el programa recursivo F

Teorema 3.4.3.-

Sea $\mathfrak{D}$ una estructura de tipo d, F un programa recursivo de tipo d y $\mathbf{F} = (\mathbf{F}_1, \ldots, \mathbf{F}_n)$ el mínimo punto fijo del operador $T_F^{\mathfrak{D}}$

Para cualquier i, $1 \leq i \leq n$, $a_1, \ldots, a_{m_i}, b \in D$

$$\mathbf{F}_i(a_1, \ldots, a_{m_i}, b) \iff \mathcal{M}_{\mathfrak{D}} \models \Pi_F X_{f_i}(a_1, \ldots, a_{m_i}, b)$$

con $\mathcal{M}_{\mathfrak{D}}$ la estructura standard de tipo td para $\mathfrak{D}$.

La demostración se consigue reconstruyendo los pasos para obtener $\mathbf{F}_i(a_1, \ldots, a_{m_i}, b)$. Como $\mathbf{F}$ es el mínimo punto fijo $\mathbf{F}_i(a_1, \ldots, a_{m_i}, b)$ se obtiene con una deducción finita de las ecuaciones de F. Al seguir esta reconstrucción en sentido contrario se obtiene la "demostración" (para Π_F) buscada , y siguiendo la "demostración" se obtiene $\mathbf{F}_i(a_1, \ldots, a_{m_i}, b)$.

Teorema 3.4.4.-

Si $\mathcal{M} = (\mathcal{T}, \mathfrak{D}, S, \text{ext}^{\mathcal{M}})$ es una estructura admisible de tipo td F es un programa recursivo de tipo d y Π_F su traducción a programa lógico simple, entonces el significado de Π_F en $\mathcal{M}$ (que como sabemos es el mínimo punto fijo definible en $\mathcal{M}$ de $T_{\Pi_F}^{\mathfrak{D}}$) se corresponde con el mínimo punto fijo definible en $\mathcal{M}$ de $T_F^{\mathfrak{D}}$.

Este teorema generaliza el anterior y establece la validez de la simulación. Los puntos fijos de $T^{\mathfrak{D}}_{\Pi_F}$ y $T^{\mathfrak{D}}_{F}$ se obtienen de manera similar y éso demuestra el teorema.

Aplicación práctica del uso de NPL a la derivación formal de propiedades de programas lógicos.

En este apartado vamos a ver dos ejemplos del uso de nuestra lógica en la verificación de programas lógicos. Nos basaremos en un trabajo de Clark y Tärnlund (7) para programas de Horn sobre interpretaciones de Herbrand. Su propuesta, que en el fondo es la propuesta de la programación lógica, consiste en que para calcular una cierta relación R sobre una estructura $\mathfrak{D}$ podemos conseguir especificar en primer orden:

- Propiedades del tipo de datos $\mathfrak{D}$ mediante una axiomatización AX_d .
- La relación R a calcular.
- Las relaciones auxiliares necesarias para obtener R.

Con esta especificación E podemos obtener dos cosas:

1) El programa lógico Π que compute R convirtiendo las dobles implicaciones que definan R en implicaciones simples separando las disyunciones en distintas cláusulas definidas (lo que puede suponer hacer lo mismo para las relaciones auxiliares).

Ejemplo:

$$\forall x \forall y(R(x,y) \leftrightarrow (x \doteq a \wedge y \doteq b) \vee S(x,f(x)))$$

$$\forall x \forall y(S(x,y) \leftrightarrow (x \doteq a \wedge y \doteq a) \vee \exists z(y \doteq f(x) \wedge S(x,z)))$$

da el programa

$$\Pi = \{\ R(a,b) \leftarrow\ ,\quad R(x,y) \leftarrow S(x,f(x))\ ,$$

$$S(a,a) \leftarrow\ ,\quad S(x,f(x)) \leftarrow S(x,y)\ \}$$

2) La verificación de propiedades del programa Π basandonos en AX_d.

Como método a seguir es interesante conseguir que $E \models \Pi$ aunque no es una condición necesaria (cfr.(24)).

Así, la lógica de primer orden la hemos usado para:

- La especificación del problema.
- La programación de Π , que resuelve el problema.
- La verificación de propiedades de Π .

es decir, todas las etapas quedan dentro de la lógica.

En NPL la verificación de programas se obtiene con los axiomas $AX = AX_t \cup AX_s \cup I_t \cup AX_e \cup AX_d$, es decir los axiomas de admisibilidad más los axiomas de tipo de datos, y el cálculo LNP. Obtendremos una propiedad φ de un programa Π si probamos $AX \overset{LNP}{\vdash} \varphi$.

Cabe plantearse que propiedades verificaremos de Π . Si Π calcula $R^{(m+1)}$, especificada como grafo de una función, las principales propiedades a varificar son análogas a la corrección parcial y a la terminación de programas imperativos convencionales:

Corrección parcial.- $\forall \bar{x} \forall y \; (\varphi_{in}(\bar{x}) \wedge \Pi R(\bar{x},y) \longrightarrow \varphi_{out}(\bar{x},y))$

donde φ_{in} decide los valores que admitimos como entradas y φ_{out} las propiedades que debe cumplir la respuesta del programa

Terminación.- $\forall \bar{x} \; (\varphi_{in}(\bar{x}) \longrightarrow \exists y \Pi R(\bar{x},y))$

El método general de prueba será utilizar un axioma de inducción sobre los datos (que debe estar disponible en AX_d) para probar la ter

minación y los axiomas de I_t, axiomas de inducción aplicados sobre la "longitud de la demostración" que hace que se cumpla $\Pi\, R(\bar{x},y)$ para probar la corrección parcial.

Si R no puede interpretarse como el grafo de una función el problema sigue en pie. Hogger en (24) ha estudiado este tema. El distingue tres tipos de propiedades a probar:

- Corrección (todo lo que calcula Π está en la relación que queremos calcular).
- Completitud (si una tupla esta en la relación que Π debe calcular, Π la computa).
- Terminación (para que tipo de objetivos el programa calcula algo).

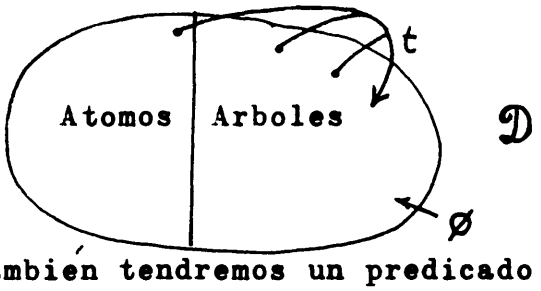
Para Hogger las dos primeras propiedades solo dependen del programa Π y de la especificación para R, mientras que la tercera puede depender de la estrategia de control que se utilice para computar un determinado objetivo. Es decir, si interpretamos los cómputos de un programa lógico como un árbol en el que cada nodo se bifurca en tantos hijos como claúsulas se puedan aplicar para intentar obtener el nodo y consideramos nodos soluciones a aquellos que solo tengan relaciones de $\mathcal{D}$ verdaderas en $\mathcal{D}$, la estrategia de control decide como se ha de recorrer ese árbol, recorrido que en ocasiones puede no ser finito y por tanto el cómputo no acaba nunca, aunque el problema pueda tener solución. Una exposición más clara y detallada puede encontrarse en (29).

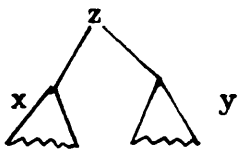
Aun siendo un tema interesante esto se aparta un poco de los objetivos del trabajo y los ejemplos que vamos a estudiar corresponden a programas con funciones interpretadas como grafos.

Ejemplo 1: Inserción en árboles ordenados.

Los árboles son una estructura de datos muy usada en programación. Como mostró Knuth (27) se pueden representar árboles generales como árboles binarios. Los árboles binarios (simplemente árboles) o son vacios o están compuestos de un átomo (que define el tipo de elementos de los árboles) y un árbol derecho y un árbol izquierdo que también son árboles.

En la estructura de datos de los árboles tendremos según esto los predicados árbol y átomo que separan en el dominio los árboles de los elementos que ocupan los nodos de los árboles. Además existe una constante $\emptyset$ para representar el árbol vacio (sin información) y una función constructora de árboles t (que toma como argumentos dos árboles x, y y un átomo z y construye el árbol t(x,z,y)





También tendremos un predicado $\in$ para indicar si un átomo está o no en un árbol. $<$ será un orden entre átomos lo que nos permitirá tener los árboles ordenados (donde el átomo raiz z de un árbol separa los elementos menores que él que se encuentran en el árbol izquier

do y los mayores que él que se encuentran en el árbol derecho, estando los dos árboles también ordenados). Estos árboles se distinguiran con el predicado ordenado.

Para formular propiedades de estos árboles distinguiremos $x,y,x_1,$ $y_1,\ldots$ como las variables para árboles y $z,z_1,z',\ldots$ como las variables para átomos. La cuantificación $\forall x\varphi$ supone $\forall x(\underline{arbol}(x)\longrightarrow\varphi)$ y $\forall z\varphi$ supone $\forall z(\underline{atomo}(z)\longrightarrow\varphi)$.

El tipo de similaridad es $d=\{\underline{árbol}, \underline{átomo}, t, \emptyset, \underline{ordenado}, \in, <\}$

Una axiomatización de esta estructura de datos es:

A1
$$\underline{arbol}(\emptyset)$$
$$\forall x\forall y\forall z\ (\underline{arbol}(t(x,z,y) \leftrightarrow (\underline{árbol}(x) \wedge \underline{árbol}(y) \wedge \underline{átomo}(z))$$

A2
$$\forall x\forall z\forall y\ (\neg \emptyset \doteq t(x,z,y))$$
$$\forall x\forall z\forall y\forall x'\forall z'\forall y'\ (t(x,z,y) \doteq t(x',z',y') \leftrightarrow x \doteq x' \wedge y \doteq y' \wedge z \doteq z')$$

(axiomas de igualdad)

A3
$$\forall z\ \neg z\in\emptyset$$
$$\forall z\forall z'\forall x\forall y\ (z'\in t(x,z,y) \leftrightarrow z' \doteq z \vee z'\in x \vee z'\in y)$$

A4
$$\underline{ordenado}(\emptyset)$$
$$\forall x\forall z\forall y\ (\underline{ordenado}(t(x,z,y)) \leftrightarrow \underline{ordenado}(x) \wedge \underline{ordenado}(y) \wedge \forall z'\ (z'\in x \rightarrow z'< z \wedge z'\in y \rightarrow z< z'))$$

A5 $\forall z\forall z'\ (z \doteq z' \vee z< z' \vee z'< z)$ (axioma del orden entre átomos)

A6 $(\varphi[\phi/x] \wedge \forall x \forall z \forall y (\varphi \wedge \varphi[y/x] \longrightarrow \varphi[t(x,z,y)/x]) \rightarrow \forall x \varphi$

(axioma de inducción para los datos)

Todos estos axiomas conforman AX_d.

La operación a realizar es insertar un átomo en un árbol ordenado de manera que el árbol resultante también esté ordenado. La relación ha definir es Insertar(t_1,a,t_2) que se cumplirá si t_2 es el árbol t_1 ordenado con el átomo a insertado.

Haciendo un análisis de los casos tenemos

1) $t_1 = \phi$ entonces $t_2 = t(\phi, a, \phi)$

2) $t_1 = t(x,b,y)$ entonces podemos distinguir tres casos según el axioma A5

(i) $a = b$ entonces $t_2 = t_1$

(ii) $a \succ b$ entonces $t_2 = t(x,b,y')$ con y' el arbol resultante de insertar a en y

(iii) dual a (ii).

Este análisis nos lleva a la siguiente especificación de la relación Insertar

Insertar$(\phi,z',t(x,z,y) \longleftrightarrow x \doteq \phi \wedge y \doteq \phi \wedge z \doteq z'$

Insertar$(t(x,z,y),z',t(x',z'',y')) \longleftrightarrow z \doteq z'' \wedge ((z \doteq z' \wedge x \doteq x' \wedge y \doteq y'$

$\vee (z \prec z' \wedge x \doteq x' \wedge$ Insertar$(y,z',y'))$

$\vee (z' \prec z \wedge y \doteq y' \wedge$ Insertar$(x,z',x')))$

De aqui obtenemos el siguiente programa lógico (donde hemos dado

a las variables de programas nombres más intuitivos)

$$\Pi = \{\ \text{Insertar}(\phi, z', t(\phi, z', \phi)) \longleftarrow\ ,$$

$$\text{Insertar}(t(x,z,y), z, t(x,z,y)) \longleftarrow\ ,$$

$$\text{Insertar}(t(x,z,y), z', t(x,z,y')) \longleftarrow z \prec z',\ \text{Insertar}(y,z',y')\ ,$$

$$\text{Insertar}(t(x,z,y), z', t(x',z,y)) \longleftarrow z' \prec z,\ \text{Insertar}(x,z',x')\ \}$$

Las propiedades a probar de Π son:

Terminación $\quad AX \models \forall x \forall z\ (\underline{\text{ordenado}}(x) \longrightarrow \exists y \Pi\ \text{Insertar}(x,z,y))$

Corrección Parcial $\quad AX \models \forall x \forall z \forall y\ (\underline{\text{ordenado}}(x) \wedge \Pi\ \text{Insertar}(x,z,y)$

$$\longrightarrow \underline{\text{ordenado}}(y) \wedge \forall z'(z' \in y \longleftrightarrow z' \doteq z \vee z' \in x))$$

La prueba de la terminación se realiza como indicamos, usando el axioma de inducción A6 para la fórmula

$$\varphi(x) = \forall z\ (\underline{\text{ordenado}}(x) \longrightarrow \exists y \Pi\ \text{Insertar}(x,z,y))$$

$\varphi[\phi/x]$

Se obtiene con unit y la cláusula definida de Π

$\text{Insertar}(\phi, z, t(\phi, z, \phi)) \longleftarrow$

$\forall x \forall z \forall y\ (\varphi \wedge \varphi[y/x] \longrightarrow \varphi[t(x,z,y)/x])$

Con las hipótesis de inducción φ, $\varphi[y/x]$ y A4 tenemos

$$AX \vdash^{\underline{\text{LNP}}} (\varphi \wedge \varphi[y/x]) \longrightarrow \forall z \forall z'(\underline{\text{ordenado}}(t(x,z,y)) \longrightarrow$$

$$\exists y_1 \Pi\ \text{Insertar}(x,z',y_1) \wedge \exists y_2 \Pi\ \text{Insertar}(y,z',y_2))$$

y distinguiendo los casos de la relación entre z y z' con A5 y usando la cláusula adecuada an cada caso más unit y conc para crear y concatenar sucesiones obtenemos

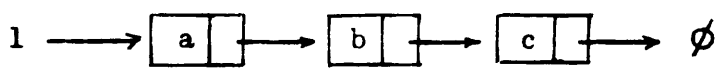
$$AX \vdash^{\underline{\text{LNP}}} \varphi \wedge \varphi[y/x] \longrightarrow \varphi[t(x,z,y)/x]$$

La prueba de la corrección parcial de Π se obtiene similarmente por inducción (axiomas de I_t) sobre i, "longitud de la demostración".

Ejemplo 2: Algoritmo Quicksort de ordenación de listas.

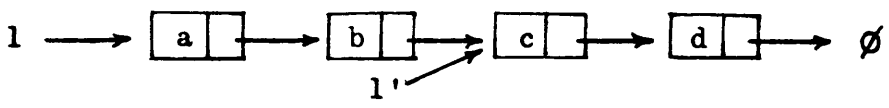
Otra conocida estructura de datos de uso habitual son las listas lineales. Estas listas se construyen a partir de la lista vacía añadiendo cada uno de sus elementos una vez. Los predicados lista y átomo separan en el dominio las listas de los átomos que las componen. Para construir listas tenemos la constante que representa la lista vacía $\emptyset$ y una función (en forma de operador) que notaremos . , que toma un átomo a y una lista l y construye la lista a.l. También tendremos el predicado de pertenencia $\in$ de un átomo a una lista; una relación de orden entre átomos $<$ y una relación de orden entre listas $\lhd$ basandose en su longitud.

Con esto podemos imaginar las listas en su representación encadenada



representa la lista l = (a.(b.(c.$\emptyset$)))

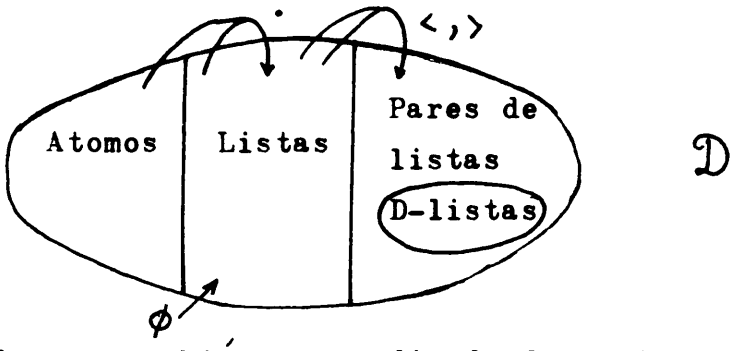
Esta situación



nos representa las listas l = (a.(b.(c.(d.$\emptyset$)))) y l' = (c.(d.$\emptyset$)) y es posible imaginar que el par de listas l,l' es la lista (a.(b.$\emptyset$))

Podemos introducir este concepto en nuestro domunio con la operacion $\langle , \rangle$ que nos da un par de listas. Pero no todos los pares de listas representan listas (en el ejemplo $\langle 1,1' \rangle$ no representa ninguna lista). El predicado d-lista nos distingirá unos de otros.

Este concepto de d-lista (listas diferencia) ha sido introducido por algunos autores y tiene la ventaja de que su utilización evita el cálculo de la concatenación de listas. Eso hace más efectivo el trabajo con ellas, como podemos ver en éste ejemplo.



Tendremos también un predicado de pertenencia a d-listas (que seguiremos llamando $\in$) y un predicado ordenada que nos indica cuando una d-lista está ordenada (los elementos de la d-lista están en orden creciente de la relacion $<$ entre átomos).

Como en el caso de los árboles llamaremos a las variables para las listas x,y,x',y',... y a las variables para los átomos z,z',..., y con el fin de simplificar la notación la cuantificación $\forall x \varphi$ ($\forall z \varphi$) supone $\forall x$ (lista(x) $\longrightarrow \varphi$) ($\forall z$ (átomo(z) $\longrightarrow \varphi$)), y en general la aparición de los pares $\langle x,y \rangle$ también supone su relativización a d-listas.

El tipo de similaridad es d = $\{$ lista, átomo, d-lista, ϕ , .,

$\langle\ ,\rangle$, $\in$ (para d-listas), $\in$ (para listas), $\prec$, $\lessdot$, <u>ordenada</u> $\}$

y su axiomatización:

<u>lista</u>$(\emptyset)$

A1

$\forall z \forall x\ (\underline{\text{lista}}(z.x) \longrightarrow \text{átomo}(z) \wedge \text{lista}(x))$

$\forall z \forall x\ \neg \emptyset \doteq z.x$

A2

$\forall z \forall x \forall z' \forall x'\ (z.x \doteq z'.x' \longleftrightarrow z \doteq z' \wedge x \doteq x')$

(axiomas de igualdad entre listas)

$\forall z\ \neg z \in \emptyset$

A3

$\forall z \forall z' \forall x\ (z \in z'.x \longleftrightarrow z \doteq z' \vee z \in x)$

(pertenencia a listas)

$\forall z \forall z'\ (z \doteq z' \vee z \prec z' \vee z' \prec z)$

A4

$\forall z \forall z' \forall z''\ (z \prec z' \wedge z' \prec z'' \longrightarrow z \prec z'')$

(axiomas del orden para átomos)

A5 $\forall x \forall y\ (x \lessdot y \longleftrightarrow (x \doteq \emptyset \wedge \exists z \exists y'\ (y \doteq z.y')) \vee \exists z \exists z' \exists x' \exists y'$

$(x \doteq z'.x' \wedge y \doteq z.y' \wedge x' \lessdot y'))$

(axiomas del orden para listas)

$\forall x \forall y\ (\underline{\text{d-lista}}(\langle x,y\rangle) \longleftrightarrow x \doteq y \vee \exists z \exists x'\ (x \doteq z.x' \wedge$

$\underline{\text{d-lista}}(\langle x',y\rangle)))$

A6

$\forall x \forall y\ (\underline{\text{d-lista}}(\langle x,y\rangle) \longleftrightarrow \exists y'\ (\underline{\text{d-lista}}(\langle x,y'\rangle) \wedge$

$\underline{\text{d-lista}}(\langle y',y\rangle)))$

$\forall x \forall y \forall y'\ (\langle x,y\rangle \doteq \langle y',y'\rangle \longleftrightarrow x \doteq y)$

A7

$\forall x \forall y \forall z \forall x' \forall y'\ (\langle z.x,y\rangle \doteq \langle z'.x',y'\rangle \longleftrightarrow (y \doteq z.x \wedge y' \doteq z'.x')$

$\vee\ (z \doteq z' \wedge \langle x,y\rangle \doteq \langle x',y'\rangle))$

$\forall z \forall x \neg z \in \langle x,x \rangle$

A8 $\forall z \forall z' \forall x (z \in \langle z'.x,x \rangle \leftrightarrow z \doteq z')$

$\forall z \forall x \forall y (z \in \langle x,y \rangle \leftrightarrow \exists y'(z \in \langle x,y' \rangle \vee z \in \langle y',y \rangle))$

(axiomas de pertenencia a d-listas)

$\forall x$ <u>ordenada</u>$(\langle x,x \rangle)$

A9 $\forall x \forall z$ <u>ordenada</u>$(\langle z.x,x \rangle)$

$\forall x \forall y$ (<u>ordenada</u>$(\langle x,y \rangle) \leftrightarrow \exists y'$(<u>ordenada</u>$(\langle x,y' \rangle) \wedge$

<u>ordenada</u>$(\langle y',y \rangle) \wedge \forall z \forall z'$ $(z \in \langle x,y' \rangle \wedge z' \in \langle y',y \rangle \leftrightarrow z \leqslant z')))$

A10 $(\varphi[\phi/x] \wedge \forall z \forall y(\varphi[y/x] \rightarrow \varphi[z.y/x])) \rightarrow \forall x \varphi$

(axioma de inducción)

A11 $(\varphi[\phi/x] \wedge \forall y (\forall y'(y' \prec y \rightarrow \varphi[y'/x]) \rightarrow \varphi[y/x]) \rightarrow \forall x \varphi$

(axioma de inducción para $\prec$).

$AX_d = \{ A1,\ldots,A11 \}$ son los axiomas para los datos.

El programa que se desea es el algoritmo Quicksort de Hoare para ordenación de listas: un átomo a de la lista que va a ser ordenada se usa para conseguir una partición del resto de átomos en dos sublistas: l' que guarda todos los elementos menores que el átomo a y l" que guarda todos los elementos mayores que a. Estas dos sublistas son tam bién ordenadas con Quicksort y conseguimos la lista general ordenada concatenando las dos sublistas con a entre las dos.

<u>Quicksort</u>$(x, \langle y,y' \rangle)$ es un predicado que ordena la lista x en la d-lista $\langle y,y' \rangle$. Usaremos una relación auxiliar <u>Partición</u>(z,x,x',x")

que divide la lista x según z en x' y x"

Un análisis de los casos nos lleva a esta especificación:

$\forall y \forall y'$ ($\underline{\text{Quicksort}}(\phi, \langle y,y' \rangle) \leftrightarrow y \doteq y'$)

$\forall z \forall x \forall y \forall y'$ ($\underline{\text{Quicksort}}(z.x, \langle y,y' \rangle) \leftrightarrow \exists y" \exists x' \exists x"$

($\underline{\text{Partición}}(z,x,x',x") \wedge \underline{\text{Quicksort}}(z.x,\langle y,y' \rangle) \wedge \underline{\text{Quicksort}}(x",\langle y",y' \rangle)$))

para $\underline{\text{Quicksort}}$, y a ésta para la relación auxiliar $\underline{\text{Partición}}$:

$\forall x' \forall x" \forall z$ ($\underline{\text{Partición}}(z,\phi,x',x") \leftrightarrow x' \doteq \phi \wedge x" \doteq \phi$)

$\forall z \forall z' \forall x \forall x' \forall x"$ ($\underline{\text{Partición}}(z,z'.x,x',x") \leftrightarrow$

($z' \leq z \wedge \exists y$ ($y \doteq z'.x' \wedge \underline{\text{Partición}}(z,x,y,x")$) $\vee$

($z < z' \wedge \exists y$ ($y \doteq z'.x" \wedge \underline{\text{Partición}}(z,x,x',y)$))

Obtenemos los programas:

Π' = { Partición$(z,\phi,\phi,\phi) \leftarrow$,

Partición$(z,z'.x,z'.y,y') \leftarrow z' \leq z$, Partición$(z,x,y,y')$,

Partición$(z,z'.x,y,z'.y') \leftarrow z < z'$, Partición$(z,x,y,y')$ }

Π = { Quicksort$(\phi, \langle y,y \rangle) \leftarrow$,

Quicksort$(z.x, \langle y,y' \rangle) \leftarrow \Pi'$Partición$(z,x,x',x")$,

Quicksort$(x', \langle y,z.y" \rangle)$,

Quicksort$(x", \langle y",y' \rangle)$ }

Verificar Π supone probar:

Terminación $AX \models \forall x \forall y \exists y' \Pi$ Quicksort$(x, \langle y',y \rangle)$

Corrección Parcial $AX \models \forall x \forall y \forall y'$ (Π Quicksort$(x, \langle y',y \rangle) \longrightarrow$

$\underline{\text{perm}}(x, \langle y',y \rangle) \wedge \underline{\text{ordenada}}(\langle y',y \rangle)$)

donde $\underline{perm}(x, \langle y', y \rangle) \leftrightarrow \forall z(z \in x \leftrightarrow z \in \langle y', y \rangle)$

La prueba de la terminación se realiza por inducción con el axioma All (de forma similar al ejemplo anterior). Es necesario probar previamente la terminación de Π';

$AX \models \forall z \forall x \exists y \exists y' \; \Pi\text{'Partición}(z,x,y,y')$

por simple inducción sobre listas (A10) en LNP.

La prueba de corrección se realiza con los axiomas de I_t. Concretamente usaremos el esquema de inducción $\underline{ind}_t'$ (cfr. P16 del lema 3.3.1) aplicado a la fórmula

$$\varphi(i) = \exists u_1,u_2,v,w \; (\underline{demostración}_\pi(u_1,u_2,v,w,i) \wedge u_1(i) \doteq x \wedge u_2(i) \doteq \langle y', y \rangle \wedge v(i) \doteq \underline{1} \wedge w(i) \doteq \underline{2}) \rightarrow \underline{perm}(x, \langle y', y \rangle) \wedge \underline{ordenada}(\langle y', y \rangle)$$

Hemos de probar

$$\forall j \; (\forall k(k < j \rightarrow \varphi[k/i]) \rightarrow \varphi[j/i])$$

Por la definición de la fórmula $\underline{demostración}_\pi$ y las propiedades de 0, 1, +, $\leq$, < tenemos

$$AX \vdash^{LNP} \underline{demostración}_\pi(u_1,u_2,v,w,j) \rightarrow \bigvee_{C \in \pi} \gamma_C(u_1,u_2,v,w,j)$$

Si C es la cláusula $Quicksort(\emptyset, \langle y,y \rangle) \leftarrow$

$$AX \vdash^{LNP} \gamma_C(u_1,u_2,v,w,j) \rightarrow u_1(j) \doteq \emptyset \wedge u_2(j) \doteq \langle y,y \rangle$$

y por la definición de perm y A9 y (T_o), (MP)

$$AX \vdash^{LNP} \gamma_C(u_1,u_2,v,w,j) \rightarrow \underline{perm}(u_1(j),u_2(j)) \wedge \underline{ordenada}(u_2(j)) \qquad (1)$$

Si C es la cláusula

$$\text{Quicksort}(z.x, \langle y,y'\rangle) \leftarrow \Pi'\text{Partición}(z,x,x',x''),$$

$$\text{Quicksort}(x', \langle y,z.y''\rangle),$$

$$\text{Quicksort}(x'', \langle y'',y'\rangle)$$

$$AX \vdash^{LNP} \gamma_C(u_1,u_2,v,w,j) \longrightarrow \exists x',x'',y'' \; (\Pi'\text{Partición}(z,x,x',x'') \wedge$$

$$\exists k \; (k < j \wedge u_1(k) \doteq x' \wedge u_2(k) \doteq \langle y,z.y''\rangle)$$

$$\exists k' \; (k' < j \wedge u_1(k') \doteq x'' \wedge u_2(k') \doteq \langle y'',y'\rangle))$$

En este punto hace falta probar ciertas propiedades de Π'. En particular su corrección parcial

$$\forall z \forall x \forall x' \forall x'' \; (\Pi'\text{Partición}(z,x,x',x'') \longrightarrow \forall z'((z' \in x' \longrightarrow z' \leqslant z) \wedge$$

$$(z' \in x'' \longrightarrow z < z') \wedge (z' \in x \leftrightarrow (z' \in x' \vee z' \in x'))))$$

que se puede probar con los axiomas de inducción I_t.

Además por el lema 3.3.2 (i) todo prefijo de una "demostración" es una "demostración". Asi para k y k' podemos aplicar la hipótesis de inducción y tenemos

$$(T_o),(MP) \quad AX \vdash^{LNP} \forall k(k < j \rightarrow \varphi[k/i]) \longrightarrow (\gamma_C(u_1,u_2,v,w,j) \wedge$$

$$u_1(j) \doteq z.x \wedge u_2(j) \doteq \langle y,y'\rangle \longrightarrow \exists x',x'',y''$$

$$(\forall z'((z' \in x' \rightarrow z' \leqslant z) \wedge (z' \in x'' \rightarrow z < z') \wedge (z' \in x \leftrightarrow z' \in x' \vee z' \in x''))$$

$$\wedge \underline{\text{perm}}(x', \langle y,z.y''\rangle) \wedge \underline{\text{perm}}(x'', \langle y'',y'\rangle) \wedge \underline{\text{ordenada}}(\langle y,z,y''\rangle)$$

$$\wedge \underline{\text{ordenada}}(\langle y'',y'\rangle)))$$

Con los axiomas para lógica de primer orden de LNP $((T_1),(MP),(G))$ y los axiomas AX_d podemos concluir

$$AX \vdash^{LNP} \forall k \; (k < j \rightarrow \varphi[k/i]) \rightarrow (\gamma_C(u_1,u_2,v,w,j) \wedge u_1(j) \doteq z.x \wedge$$

$$u_2(j) \doteq \langle y,y'\rangle \longrightarrow \underline{\text{perm}}(z.x, \langle y,y'\rangle) \wedge \underline{\text{ordenada}}(\langle y,y'\rangle)) \qquad (2)$$

demostrando la propiedad de que dos particiones ordenadas en el sentido visto se pueden concatenar dando una d-lista ordenada.

De (1) y de (2) con (T_o),(MP) y la definición de la fórmula demostración $_{\Pi}$ podemos concluir

$$AX \vdash^{LNP} \forall j\ (\forall k\ (k < j \rightarrow \varphi[k/i]) \rightarrow \varphi[j/i])$$

y por tanto

$$AX \vdash^{LNP} \forall i\ \varphi$$

que con el axioma (Π) equivale a

$$AX \vdash^{LNP} \forall x \forall y \forall y'\ (\Pi\ Quicksort(x, <y', y>) \rightarrow \underline{perm}(x, <y', y>) \wedge \underline{ordenada}(<y', y>))$$

BIBLIOGRAFIA:

(1) Andréka, H., I. Németi, I. Sain, Completeness results in verification of programs and programs schemes, en: MFCS'79, Lectures Notes in Computer Science 74 (1979), Springer Verlag, 208 - 218.

(2) Andréka, H., I. Németi, I. Sain, A Complete Logic for Reasoning about Programs via Nonstandard Model Theory, partes I y II, Theor. Comp. Science 17 (1982), 193 - 212, 259 - 278.

(3) Apt, K.R., M. H. van Emdem, Contributions to the Theory of Logic Programming, JACM 29 (1982), 841 - 862.

(4) Burstall, R. M., Program Proving as Hand Simulation with a Little Induction, Information Processing'74, North Holland, 308 - 312.

(5) Cartwright, R., Nonstandard fixed - points, Proc of Logics of Programs, Workshop, Carnegie - Mellon Univ., Lectures Notes in Computer Science (1983),86 - 100.

(6) Cartwright, R., Recursive Programs as Definitions in First Order Logic, SIAM Journal of Computing.

(7) Clark, K. L., S.-A. Tärnlund, A First Order Theory of Data and Programs, Proc. IFIP Congress'77, North Holland (1977), 939 - 944.

(8) Colmerauer, A., K. Kanoui, P. Roussel, P. Pasero, Un Sisteme de Communication Homme - Machine en Francais, Groupe de Recherche de Intelligence Artificielle, Universite d'Aix - Marsella (1973).

(9) Cook, S. A., Soundness and completeness of an axiom system for program verification, SIAM J. Comp. 7 (1978), 70 - 90.

(10) Dijkstra, E., A Discipline of Programming, Englewood Cliffs, N.J. Prentice Hall (1976).

(11) van Emdem, M. H., R. A. Kowalski, The Seantics of Predicate Logic as a Programming Language, JACM, 23 (1976) 733 - 742.

(12) Engeler E., Algorithmic Properties of Structures, Math. Syst. Theory 1,(1967) 183 - 195.

(13) Fischer, M. J., R. E. Ladner, Propositional Modal Logic of Programs, Proc. 9th Ann. ACM Symp. of Theory of Computing (1977) 286-294.

(14) Floyd, R. W., Assigning Meanings to Programs, Proc. AMS Symp. Applied Math. 19, American Math. Society Providence, R.I. (1967) 19 - 31.

(15) Gabbay, D., A. Pnueli, S. Shelah y J. Stavi, On the Temporal Analysis of Fairness, Proc. 7th ACM Symp. Principles Prog. Lang.(1980), 163 - 173.

(16) Green, C., The aplicattion of theorem proving to question - answering systems, Ph. D. Thesis, Computer Science Dept. Standford University (1969).

(17) Hájek, P., Making Dynamic Logic First - Order, Mathematical Foundations of Computer Science (Proc. Strbske Pleso 1981), Lectures Notes in Computer Sciences 118, Springer Verlag (1981) 287 - 295.

(18) Harel, D., First - Order Dynamic Logic, Lectures Notes in Computer Science 68, Springer Verlag (1979).

(19) Harel, D., Proving the Correctness of Regular Deterministic Programs : A unifying Survey using Dynamic Logic, Theor. Computer Science 12, North Holland (1980), 61 - 81.

(20) Harel, D., Dynamic Logic, Lectures Notes, Department of Applied Math., The Weitzman Institute of Science, Rehorot, Israel (1982).

(21) Hitchock, P., D. Park, Induction Rules and Termination Proofs, M. Nivat (ed.) Automata, Languages and Programming, Noth Holland (1973).

(22) Hoare, C.A.R., An Axiomatic Basis for Computer Programmings, Comm. Assoc. Comput. Mach. 12 (1969) 576 - 583.

(23) Hogger, C. J., Derivation of Logic Programs, JACM 28 (1981) 372 - 422.

(24) Hogger, C. J., Introduction to Logic Programming, Academic Press (1984).

(25) Hortala, M. T., M. Rodriguez A., Hoare's Logic for Nondeterministic Regular Programs : A Nonstandard Completeness Theorem, Prepint, D. E. F., U. Complutense, Madrid (1985).

(26) Keisler, H. J. , Model Theory for Infinitary Logic, North Holland, (1971)

(27) Knuth, D., The Art of Computer Programming, vol 1 - 3, Addison Wesley Reading (1968/1973).

(28) Kowalski, R. A., Predicate Logic as a Programming Language, IFIP'74 (1974) 569 - 574.

(29) Lloyd, J. W., Foundations of Logic Programming, Springer Verlag (1984).

(30) McCarthy, J., A Basis for a Mathematical Theory of Computation, en P. Braffort y P. Hirschberg (eds.), Computer Programming and Formal Systems, North Holland (1967) 33 - 70.

(31) Mirkowska, G., On formalized systems of algorithmic logic, Bull. Acad. Sci. Ser. Sci. Math. Astr. Phys. 22 (1974) 421 - 428.

(32) Naur, P., Proof of algorithms by general snapshuts, BIT 6 (1966) 310 - 316.

(33) Németi, I., Nonstandard Dynamic Logic, Logic of Programs (Proc. New York'81), Lectures Notes in Computer Science 131, Springer Verlag, (1982) 331 - 348.

(34) Parikh, R., Propositional Dynamic Logic of Programs: A Survey, Workshop on Logic of Programs (E. Engeler ed.), Lectures Notes in Computer Science 125, Springer Verlag (1984), 102 - 144.

(35) Pnueli, A., The Temporal Logic of Programs, Proc. 18th IEEE Symp. Found. Comp. Science (1977) 45 - 57.

(36) Pratt, V. R., Semantical Considerations on Floyd - Hoare Logic, Proc 17th IEEE Symp. Found. Comp. Science (1976) 109 - 121.

(37) Roussel, P.,PROLOG: Manuel de reference et d'utilisation, Groupe d'Intelligence Artificielle, UER de Luming. (1975), Marsella.

(38) Sain, I., Structured Nonstandard Dynamic Logic, Zeitschr. für Math. Logick u. Grund. des Math. Heft 3, Band 30.(1984).

(39) Salwicki, A., Formalized Algorithmic Languages, Bull. Acad. Polon. Sc. Ser. Sci. Math. Astron. Phys. 18 (1970) 227 - 232.

(40) Schmitt, P. H., Diamond Formulas. A fragment of Dynamic Logic with Recursively Enumerable Validity Problem, Prepint. Mathematisches Institut Universität Heilderberg.

(41) Sickel, S., Specification and derivation of Programs, M. Brown y G. Schmidt (eds.), Theoretical Foundations of Programming Methodology (1984) 103 - 133.

(42) Tärnlund, S.-A., An axiomatic data base theory, Logic and Data Bases, M. Gallaire y J. Minker (eds.), Plenum Press (1978) 259 - 289.

(43) Tarski, A., A Lattice - Theoretical Fixpoint Theorem and its Applications, Pacific J. Math. 5 (1955) 285 - 309.

(44) Turing, A., Checking a Large Routine, en Report of a Conference on High Speed Automatic Calculating Machines, Univ. Math. Lab. Cambridge (1949) 67 - 69.